UNIVERSIDAD DE IXTLAHUCA CUI, A. C.

LICENCIATURA EN INGENIERÍA EN TELECOMUNICACIONES, SISTEMAS Y ELECTRÓNICA

INCORPORADA A LA
UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
Clave 8968-66

Diseño e implementación de un filtro de Kalman para un controlador PID acoplado a un invernadero.

TESIS

QUE PARA OBTENER EL TÍTULO DE

INGENIERO EN TELECOMUNICACIONES, SISTEMAS Y ELECTRÓNICA

PRESENTA

ERNESTO CORONA SÁNCHEZ
ASESOR: M. EN C. JESÚS NAMIGTLE JIMÉNEZ

IXTLAHUACA, MÉXICO. ABRIL, 2025

Revisores

M. en A. T. I. Karina
Balderas Pérez

Dr. en C. Jorge Edmundo
Mastache Mastache

M. en C. Jesús
Namigtle Jiménez

Dr. en Edu. Bernardo
Juárez González

Ing. Hugo Ángeles Cruz

Índice

Índice de Figuras

Índice de Tablas

Agradecimientos

Agradezco infinitamente a mis padres, quienes me apoyaron en cada momento de mi vida por más complicado que fuera, por todos los esfuerzos que hicieron para darme todo lo que necesite, por enseñarme a hacer las cosas por mi mismo, por impulsarme a ser mejor día a día, los amo. A dios, que me ha dado mucha paciencia, salud y bienestar para llevar mi vida con plenitud, por prestarme la fuerza para levantarme en los malos momentos y nunca darme por vencido.

A mi asesor, el M. en C. Jesús Namigtle Giménez por aceptarme como su tesista, guiarme y brindarme su tiempo para aprender y mejorar. Gracias por motivarme a seguir con este proceso, este trabajo es gracias a usted y le agradezco su confianza para desarrollar este proyecto.

A cada uno de mis docentes, que me mostraron una parte de ellos con su conocimiento y su trato como ser humano, mi completa admiración porque lo que aprendí se lo debo a todos y cada uno de ustedes y es algo de lo que les estaré profundamente agradecido.

Un especial agradecimiento a la Universidad de Ixtlahuaca CUI, por dar la oportunidad de titular a personas como yo que vio cumplir el sueño de completar una carrera conociendo en el camino a personas maravillosas.

Resumen

En México, el campo de la agricultura protegida ha crecido exponencialmente en los últimos años. Esto se debe a la necesidad de cultivar alimentos bajo condiciones controladas que mitiguen los problemas generados por el cambio climático, la presencia de plagas y enfermedades, entre otras amenazas del cultivo tradicional. Para establecer estas condiciones en los invernaderos, generalmente se implementan sistemas de control. Una de las variables más importantes a regular es la temperatura, ya que su precisión es crucial para el buen desarrollo de los cultivos.

En los sistemas de control el procesamiento de señales es fundamental, por lo que se requiere una etapa de filtrado para eliminar ruido y perturbaciones que puedan afectar la regulación de variables. Por esta razón, en esta tesis se analiza el filtro de Kalman discreto como una alternativa para mitigar el ruido en un sensor de temperatura y mejorar el rendimiento de un controlador PID. Se desarrolla un sistema de monitoreo que, además de supervisar la temperatura, la humedad del suelo y la luminosidad, integra un control digital de temperatura junto con el filtro de Kalman.

Se presenta la sintonización del controlador PID mediante el método experimental de Ziegler-Nichols, así como la implementación de los scripts en lenguaje Python para el sistema de monitoreo, el controlador y el filtro. Finalmente, se analiza el desempeño del control de temperatura mediante métricas como la varianza y medidas en el dominio del tiempo, incluyendo tiempo de retardo, tiempo de subida, sobreelongación y tiempo de asentamiento.

Capítulo 1

Planteamiento del problema y propuesta de solución

En este capítulo se analizará el contexto actual de la automatización de procesos en el país, haciendo hincapié en el papel que desempeñaran a futuro. Se plantea una propuesta de un sistema de monitoreo local para un invernadero de dimensiones reducidas con el objetivo de manipular la temperatura interna utilizando un controlador Proporcional-Integral-Derivativo (PID). Para mejorar su rendimiento a través de la integración de un filtro digital de características adaptativas y óptimas.

1.1. Introducción

Actualmente los sistemas de control desempeñan un papel importante en distintos sectores de la industria ya que las características que los distinguen representan más de un beneficio en cuanto al control de procesos, en la optimización de recursos, la generación de ingresos de una compañía, manteniendo una baja taza de error por la mínima intervención humana, Dichos sistemas son ampliamente utilizados en distintas ramas del sector industrial. En el contexto nacional, se conoce a través de un reporte realizado por parte de la NMSC (Next Move Strategy Consulting) del impacto que se ha tenido mediante una estimación del tamaño del mercado de automatización de procesos industriales en México, en el cual se valoró en 2.27 mil millones USD tan solo en el año 2023, y se prevé que para 2030 aumente casi al doble (4.11 mil millones USD aproximadamente) [1].

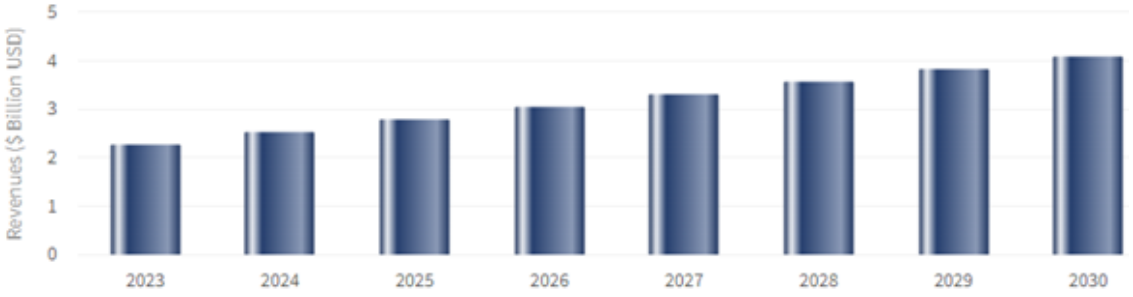


Figura 1.1: Ingresos del mercado de automatización de procesos industriales en México 2023-2030 [1].

La automatización de procesos tiene la finalidad de presentar un beneficio a la sociedad, no solo a nivel industrial dado que los procesos económicos pueden adecuarse y optimizarse de mejor manera a través de herramientas informáticas, de hardware y software que apoyen en la ejecución de tareas de manera autónoma con el objetivo de proporcionar tecnologías capaces de estructurar un sistema de control eficaz, de supervisión y optimización. Los sensores, los PLC's, los sistemas SCADA (sistemas de control de supervisión y adquisición de datos) así como las interfaces HMI (hombre-máquina) son comúnmente asociados con los sistemas de automatización, consecuentemente esto lleva a una demanda en el crecimiento de la tecnología con el fin de reducir costos, mitigar errores, optimizar procesos para obtener mayor precisión y productividad. En el contexto de la agricultura moderna y la gestión de invernaderos, la precisión en el control de variables ambientales es fundamental para optimizar la producción de cultivos. En este sentido, el controlador proporcional-integral-derivativo (o también conocido como PID) en conjunto con un filtro de Kalman se presenta como una solución tecnológica avanzada que puede contribuir a la forma en que se controlan las condiciones térmicas en los invernaderos presentando así una alternativa de automatización. Este enfoque no solo permite una estimación precisa de estados ambientales, sino que también proporciona una adaptabilidad única ante cambios en las condiciones, lo que se traduce en una producción agrícola más eficiente y sostenible. En el presente trabajo, se realiza la propuesta de un sistema de monitoreo con funciones intuitivas, consultando primeramente trabajos relacionados

en años recientes. Esto implica la revisión de metodología, componentes electrónicos, tipos de control para la temperatura, circuitos asociados, entre otros. Con el fin de seleccionar los componentes adecuados para la estructuración del sistema tanto en el hardware como el software. El diseño del sistema se encuentra seccionado de la siguiente manera: Simulación de circuitos para la comunicación de la Raspberry Pi y GrovePi+ con el actuador térmico para sincronizar y administrar la energía eléctrica según el algoritmo de control PID con respecto al sensor. Desarrollo del filtro de Kalman, realizando en primera instancia simulaciones con lecturas de la curva de reacción de la temperatura interna del invernadero para posteriormente adicionarlo al algoritmo del control PID digital desde el entorno de programación Thonny IDE. Desarrollo de las interfaces gráficas de usuario que conforman al sistema de monitoreo, determinando la estructura de las funciones; grabado de datos, control PID y visualización de gráficos de las variables físicas. Teniendo estas etapas, los componentes se conectarán entre sí para formar un sistema el cual permita recopilar los datos, procesarlos en conjunto con el controlador y el filtro, para un análisis gráfico con apoyo de Matlab además de calcular el error promedio en tiempo estable así como otras métricas en el dominio de tiempo disponible en la sección de resultados.

1.2. Problemática

Un controlador PID tiene el objetivo de mejorar el comportamiento de un sistema con el propósito de alcanzar y mantener un valor deseado de acuerdo al control de una variable en específico de tal manera que se logre mitigar el error resultante de la diferencia entre la señal real de salida con respecto al valor deseado. De acuerdo a la variable a controlar es imprescindible contar con sensores que permitan conocer su comportamiento a través del tiempo. Sin embargo, uno de los fenómenos que llegan a afectar su funcionamiento es la presencia de ruido, considerada también una perturbación capaz de alterar las lecturas de la variable física. Por lo general, para solventar en gran medida los errores de medición se aplica algún tipo de filtro, comúnmente se encuentran los filtros pasa bajos ya que no es complejo de implementar y resulta responder de buena manera. Por otra parte, el uso de filtros tradicionales como FIR o IIR digitales al ser incapaces de distinguir el tipo de ruido que altera las lecturas del sensor no logran ajustarse automáticamente si las características de la perturbación cambian con el tiempo, perdiendo así su efectividad. Los sensores son propensos al ruido dependiendo del proceso y de las condiciones a las que está sometido. El ruido provoca problemas en el rendimiento del control PID generando sobreajustes, oscilaciones durante un tiempo prolongado y por ende cambios bruscos en la señal de control que rige al actuador desestabilizando la salida del sistema. En años recientes se han implementado técnicas de control de temperatura en espacios reducidos y en invernaderos de dimensiones pequeñas no es la excepción. En estos trabajos por lo general se lleva a cabo la implementación de controles ON-OFF, PD, PI y PID con distintos modelos de sensores, diferente resolución y sensibilidad, en donde, o utilizan algún tipo de filtro tradicional para mitigar errores de medición o bien se omite y trabajan con la señal contaminada. Por ejemplo, en 2022 se propuso un sistema de control y monitoreo para un invernadero cuyos componentes se comunican con Arduino Uno, el tipo de control consta de encender y apagar dispositivos

para mantener los valores deseados [27]. La implementación de estrategias de control más robustas es necesario dado que el control ON-OFF presenta ciertas desventajas como el desgaste de componentes mecánicos y la ineficiencia energética. Es por estos factores que se requiere de un sistema de control de temperatura confiable que además de la supervisión, el control deba ser lo suficientemente preciso para un mejor rendimiento.

1.3. Propuesta

En el presente trabajo, se tiene el propósito de diseñar e implementar un filtro de Kalman que actué sobre la lectura de un sensor de temperatura, para que junto a un controlador PID enfocado a la misma variable se integre a un sistema de monitoreo en un invernadero de dimensiones reducidas como prototipo. Tanto el controlador como el filtro trabajaran de manera conjunta en tiempo real para lograr una alta precisión, ambos de manera digital y dependiente de su accionamiento por medio del usuario a través de interfaces gráficas. Dado que se realizará la codificación del controlador y el filtro, se contempla el uso de la tarjeta de desarrollo Raspberry Pi adicionándole el *hat* (también conocido como extensión, complemento o armazón) GrovePi+. Raspberry Pi opera en el entorno de Raspbian, específicamente en su distribución "*Raspbian for Robots*", por lo que es posible aprovechar el lenguaje de programación Python desde el entorno de programación Thonny IDE. GrovePi+ contiene una serie de puertos (desde digitales, analógicos, I2C, etc.) y funciona exclusivamente con "*Raspbian for robots*" dado que este sistema operativo alberga una librería en específico permitiendo la manipulación de todos sus puertos por medio de instrucciones basadas en Python. Además del sensor de temperatura (que es indispensable para el funcionamiento del PID), se plantea el uso de un sensor enfocado a la medición del nivel de la luz y otro más destinado a la humedad de suelo, esto con el fin de acondicionar el invernadero teniendo en cuenta otras variables básicas que comprende a un invernadero de interior. Los sensores utilizados son parte de la familia "*grove*", elegidos dada su compatibilidad con el *hat* así como por el bajo margen de error (teórico) que poseen. El controlador y el filtro son parte del sistema de monitoreo intuitivo, el sistema está compuesto por interfaces gráficas de usuario las cuales se diferencian por la función que realiza cada una de ellas. La interfaz gráfica principal muestra información captada por cada uno de los sensores y presenta un menú de 4 opciones; iniciar el PID, mostrar la ventana de gráficos de las variables físicas, una herramienta de almacenamiento de datos y una opción de fin de ejecución del sistema. Todo esto con el propósito de monitorear las condiciones ambientales internas del invernadero de manera local. Se realizará el diseño de circuitos electrónicos necesarios para la comunicación del controlador digital con el actuador para llevar a cabo el cambio de la temperatura, para ello se usa el programa *Proteus 8 Professional* y *LTspice* para realizar simulaciones y comprender de mejor manera su funcionamiento. Los circuitos electrónicos son dos, el primero que tiene la función de emitir pulsos de 4.7V cada vez que la señal eléctrica de la toma de 120V AC (senoidal) cruce por cero, con el fin de que el pulso PWM emitido por la Raspberry Pi logre sincronizarse correctamente. Para ello se utiliza el circuito integrado H11AA2X. El segundo circuito esta 100 % enfocado con el control de ener-

gía que obtendrá el actuador, cuyos componentes principales son el optoacoplador MOC3021, el Triac BT139 y el amplificador operacional TL084CN. Se elaborará la PCB de cada uno de los circuitos, el circuito de cruce por cero requiere como entrada la señal de 120V AC mientras que para el segundo circuito depende de la señal PWM de la Raspberry Pi así como de dos fuentes independientes con salida de 12V DC para una etapa de amplificación. La manera de comprobar su funcionamiento antes de la integración con el sistema se lleva a cabo con el uso de un osciloscopio digital, en el que se observara cada una de las señales en la salida de los circuitos. Se integra el sistema de monitoreo en su totalidad, esto es, conexión de sensores con Raspberry Pi y GrovePi+ en sus respectivos puertos, interconexión de los circuitos electrónicos, conexión de fuentes de voltaje, así como su organización y colocación en el invernadero. Al tener toda la parte de hardware preparada, es desde el entorno de Thonny IDE que es posible ejecutar el sistema de monitoreo, navegar entre las interfaces y realizar pruebas con los distintos *setpoints* (valores de referencia) según el nivel de temperatura deseado.

1.4. Justificación

El control PID puede ser aplicado para distintos procesos, no solo en invernaderos. Una parte fundamental a la hora de su implementación está en la precisión de los sensores, dependiendo de la variable a medir y el modelo pueden presentar problemas de medición. Los filtros de Kalman en los últimos años han tenido presencia en la estimación de estados para procesos como en gasificadores, en donde el filtro se usa para estimar estados no medidos en función de los datos disponibles del sensor, en base con los resultados obtenidos demostró ser una herramienta práctica para el monitoreo y estimación en tiempo real [28]. Aplicar el control PID y el filtro de Kalman a presentado una mejora en el rendimiento de la estimación de parámetros en condiciones con presencia de ruido en bucle cerrado. En trabajos recientes también mostró que la colaboración del filtro con el control PID no necesita de un modelo del sistema en concreto [29]. El control de la temperatura, humedad de suelo, la luz, son algunas de las variables que en los invernaderos se deben monitorear y controlar, con el fin de cultivar alimentos fuera de temporada y de buena calidad. Este sistema contribuirá principalmente al público de la sociedad mexicana debido a que el sistema de control empleado para el prototipo del invernadero está diseñado y desarrollado para el cultivo de interior, estructurado con materiales de fácil acceso permitiendo obtener un desarrollo óptimo controlando la temperatura interna, de manera que el usuario tenga como prioridad la supervisión de la planta por medio de interfaces gráficas amigables. Raspberry Pi no requiere de licenciamiento, por lo que es posible mejorar a futuro el sistema en general, su precio accesible, la compatibilidad con otros softwares (libres), el bajo consumo energético y su versatilidad la colocan como una de las mejores alternativas para el desarrollo de sistemas de control. El sistema de monitoreo propuesto tiene pilares fundamentales que radican en la implementación del sistema en Python, dado que es posible aprovechar el lenguaje de programación, el lenguaje no solo ofrece una amplia gama de bibliotecas, herramientas para el procesamiento de datos y algoritmos de control, también es posible encontrar herramientas para la creación de interfaces gráficas de usuario in-

tuitivas y amigables, en donde los usuarios necesitan interactuar de manera sencilla con el sistema para tomar decisiones informadas y precisas. En los últimos años, la salida de diferentes placas de programación ha representado un gran impacto en el desarrollo de proyectos enfocados a distintas áreas, en el campo de la robótica, la automatización de procesos relacionados con la agricultura, en el diseño de sistemas de control, domótica, internet de las cosas, adquisición y procesamiento de datos, etc. Demostrando la versatilidad que pueden llegar a ofrecer según las características de cada una. Desde la salida de Arduino (en el año 2005), Raspberry Pi (en el año 2012) y otras placas que, si bien no han tenido el mismo alcance en el mercado, como es el caso de Beaglebone o Intel Edison, la elaboración y publicación de proyectos de automatización han crecido notablemente con estas tarjetas como parte fundamental en el área del control. Es en Arduino que se han encontrado propuestas de monitoreo y control de condiciones climáticas en invernaderos cuyos componentes (como los sensores) se adecúan de acuerdo a la compatibilidad que tienen con las tarjetas. GrovePi+ permite que Raspberry Pi acceda directamente a algunos sensores grove puesto que Raspberry Pi posee un bus I2C y un bus serie. Estos últimos pueden conectarse directamente a los sensores a través de los puertos I2Cy el puerto USART. Este es un *hat* utilizado en algunos trabajos para elaboración de prototipos de estaciones meteorológicas a nivel didáctico [30], a formado parte de distintos proyectos como el sistema de monitoreo de calidad de aire implementado con IoT del cual se logró supervisar con eficacia el aire de manera remota y con ello plantear la posibilidad de cubrir una mayor área [31], en la integración de módulos con el fin de controlar el riego de forma inteligente [32] por mencionar algunos ejemplos. Dada su versatilidad, GrovePi+ es buen complemento de Raspberry Pi para comunicar elementos electrónicos de control y monitoreo en la elaboración de nuevos proyectos para la solución de distintos problemas.

1.5. Antecedentes

En este apartado se presentan algunos de los trabajos que se relacionan con el desarrollo de esta tesis y por consecuencia con el sistema propuesto, haciendo hincapié en las características, estructuras, métodos y sobre todo, los resultados a los que los diversos autores lograron llegar formando una base sólida para la implementación del controlador PID dentro del sistema de monitoreo en el invernadero. En el ámbito del desarrollo de sistemas de monitoreo, es por medio de la Raspberry Pi que se ha podido encontrar soluciones en conjunto con módulos IoT sistemas modulares enfocados al control de motores, implementación de interfaces con diferentes funciones, escalable y con un buen manejo de datos con el uso de API's en la nube para su monitoreo remoto [33]. En este trabajo al igual que el sistema propuesto en esta tesis, se usa la Raspberry Pi como principal componente para albergar el sistema de monitoreo, sin embargo, se logra encontrar puntos destacados; T. J. E. Andrés [33] realiza el control de la temperatura interna del invernadero a través del uso de ventiladores, dependiendo completamente de la generación térmica a través del sol por lo que no se usa un método fiable que determine el completo control de la variable física. Otro de los puntos a tomar en cuenta de este trabajo es la confiabilidad de los sensores ya que al no estar completamente calibrados no

se asegura la precisión de las lecturas, de allí que el filtro de Kalman sea indispensable en el controlador propuesto en esta tesis. En cuanto al rendimiento, T. J. E. Andrés [33] presenta que, a pesar de los recursos que utiliza para su sistema como la base de datos, la comunicación de la Raspberry Pi con plataformas IoT como lo es *Thingspeak*, manejo de sensores y actuadores, obtuvo resultados alentadores ya que no se rebasa el 5 % del uso de la CPU demostrando que Raspberry Pi se puede adecuar para la implementación del sistema de monitoreo, el controlador y las interfaces de manera local. En el año 2021, se propuso realizar un sistema cuyo controlador principal es Raspberry Pi 3 destinado a un invernadero casero, las variables a controlar son las mismas que se han descrito en trabajos similares, sin embargo, el modo de control consta de la técnica de control ON-OFF dependiendo de umbrales para cada variable y monitoreado a distancia por medio de *Thingspeak*, logran llevar a cabo tareas de control de luz, temperatura por medio de extractores y es posible monitorear ininterrumpidamente desde la nube [34]. Se pueden seguir encontrando otros tipos de sistemas de monitoreo y/o automatización en invernaderos con distintas tecnologías, con propósitos similares y técnicas más especializadas que otras. Uno de los ejemplos más claros puede verse en sistemas que auxilian a la toma de decisiones en tiempo real [35], en donde, los sensores también juegan un rol importante para mantener controlados los puntos de referencia de las variables climáticas. En este trabajo se logra identificar el método de control de la temperatura, el cual es por medio de un controlador difuso cuyos elementos principales son las ventanas y válvulas de riego. El rango de funcionamiento va desde los 14° C para su calefacción y 25° C para su ventilación (enfriamiento). Este controlador, es apoyado mediante un DSS (sistema de apoyo a la toma de decisiones) basado en tres etapas; supervisión, control y estrategias. Además de Raspberry módulos como *Green Tech control* son capaces de regular la temperatura y la humedad con distintos modos de control con el objetivo de asegurar la calidad de las cosechas en base a un control de riego [36]. Las variables que contempla este trabajo van desde la temperatura, luz, humedad, CO2 y PH. El tipo de control se puede llevar a cabo de 3 maneras; control PD, arboles de decisión y también por medio de control de tiempo, de las cuales la que más interesa para el desarrollo de la presente tesis es el primera dado que los últimos dos métodos requieren de la intervención del usuario (manual). A pesar de no manejar una interfaz gráfica para el manejo del sistema, el método de control difuso PD mantiene una similitud con el PID propuesto en mi sistema de monitoreo, y es la utilidad y facilidad de implementación sin requerir de un modelo matemático para el control de la temperatura interna. En 2020. se presenta el diseño de un invernadero implementando métodos de control de humedad y de temperatura. Para ello se hace uso de una interfaz gráfica con Labview para la adquisición de datos, almacenamiento en archivos excel y su posterior presentación con ayuda de la plataforma Thingspeak para su monitoreo remoto. En cuanto al control de la temperatura interna se implementa el sensor LM35 y un control tipo PID, para este último hace uso de distintos métodos para la obtención de ganancias KP, KI y KD; uso del sintonizador de Matlab con Simulink para el calentamiento y el método de Ziegler-Nichols para control de enfriamiento principalmente. Los resultados obtenidos se basan en los valores teóricos con respecto a la salida real del sistema con una coincidencia del 87.65 % para el control de calefacción y un 77.29 % para

el control de enfriamiento. El sistema de monitoreo resulto ser eficaz con bajo nivel de ruido en los valores adquiridos [37]. El sistema de monitoreo planteado en esta tesis tiene una diferencia en cuanto al uso del método de Ziegler-Nichols ya que será utilizado completamente para la calefacción interna del invernadero, por lo que, si es necesario bajar la temperatura se recurrirá al mismo efecto del controlador PID. El uso de tarjetas como Arduino se adaptan bien para este tipo de aplicaciones a pesar de no ser utilizado completamente en el sector industrial, es posible obtener resultados alentadores. Tres años después se diseña y desarrolla una incubadora de monitoreo y control en tiempo real de forma experimental desde la recopilación de datos, determinación del controlador teniendo en cuenta la Raspberry Pi en su modelo 3 B utilizando los pines GPIO para la interconexión de sensores y actuadores en la cual es a través de la obtención de los datos que se llega a un diseño para su implementación. Se elaboran circuitos para adecuar la conexión de los elementos del sistema de control, así como el desarrollo de software para su manipulación vía remota desde una interfaz de usuario. Obtienen resultados confiables en cuanto al control PID que se realiza sobre la temperatura interna de la incubadora teniendo como único setpoint un valor de 37° C en un tiempo aproximado de 20 minutos con un error de ±0.2 ° C [38]. El autor realiza una interconexión de componentes teniendo a la Raspberry Pi como principal controlador, en él se comunica tanto la parte del software como el hardware sin la necesidad de un hat, caso contrario al sistema de monitoreo propuesto en el presente trabajo ya que GrovePi+ tiene mayor participación en la interconexión de sensores y circuitos. Las interfaces gráficas de usuario juegan uno de los papeles fundamentales en estos tipos de sistemas ya que es a través de ellas que se ejecutan las funciones principales, siendo la librería Tkinter una herramienta confiable en el procesamiento de los datos y en el control de variables físicas como lo es la temperatura. El uso del filtro de Kalman en los sistemas de monitoreo supone una alternativa para la obtención de mejores lecturas con bajo margen de error, en el trabajo de I. Firmansyah et al. [39] se observó un sistema de monitoreo de temperatura integrando un horno para el controlador PID digital, teniendo como controlador una placa de Arduino Yun y sensores de termopar. En su implementación, el filtro de Kalman logro reducir el ruido de los valores provenientes del sensor de temperatura en tiempo real. El sistema de monitoreo remoto funciona mediante acceso web con resultados exitosos. Se concluye la buena efectividad del sistema de monitoreo de temperatura por medio de wifi y su posibilidad de utilizarlo para diferentes propósitos [39]. Este es uno de algunos trabajos en los que se logra adaptar el filtro de Kalman en tiempo real para mejorar el funcionamiento del controlador haciendo más preciso el manejo de la temperatura interna de manera digital. A pesar de usar como controlador principal una tarjeta de Arduino, es posible implementarse el filtro y el controlador con Python. Z. Gao et al. [40] proponen un sistema de control de temperatura para invernaderos, el diseño del control PID se basa en el filtro de Kalman simulando sus efectos en las lecturas del sensor con Matlab lo cual le sirve para demostrar la respuesta de manera rápida, así como su convergencia y su estado estacionario mejorando considerablemente la manipulación de la variable física. Los resultados del filtro denotaron que, mientras la ganancia de Kalman decrece, los valores de predicción son más precisos [40]. Dado que los autores realizan un análisis mediante la simulación del controlador PID y

del filtro, se puede realizar un comparativo con el sistema de monitoreo propuesto en este trabajo ya que se maneja un nivel de error de 0.2 con el uso del filtro y sin él se eleva hasta 1.37 indicando bajo rendimiento del control de la temperatura. En 2023, se llevó a cabo el diseño e implementación de un controlador PID en conjunto con el filtro de Kalman montado en una Raspberry Pi 4 destinado a sistemas de riego capilar fibroso superficial debido al bajo rendimiento de los métodos de riego existentes. Le denominan control KF-PID y tiene el objetivo de minimizar errores de estimación para mejorar la gestión del agua y lograr un mayor índice de productividad. El control KF-PID tuvo mejor rendimiento que el control difuso empleado tanto en simulación con ayuda de la herramienta de Simulink (Matlab) como en la implementación física [41]. Es debido a los resultados del filtro en esta colección de trabajos que el filtro de Kalman será parte fundamental en la implementación del controlador propuesto en la presente tesis.

1.6. Hipótesis

La implementación de un filtro de Kalman en la lectura de un sensor grove de temperatura v1.2 mejorará el desempeño de un controlador PID en un invernadero de pequeñas dimensiones.

1.7. Objetivo general

Diseñar e implementar un filtro de Kalman en un controlador PID de temperatura dentro de un sistema de monitoreo aplicado a un invernadero de pequeñas dimensiones.

1.8. Objetivos específicos

- Analizar e implementar un filtro de Kalman en tiempo discreto para la lectura de un sensor de temperatura.
- Diseñar, construir y acondicionar un invernadero de pequeñas dimensiones.
- Instrumentar los sensores grove: de temperatura v1.2, humedad de suelo y de luz.
- Diseñar y desarrollar un sistema de monitoreo por medio de interfaces gráficas de usuario para la supervisión de temperatura, luz, y humedad de suelo.
- Implementar el sistema de monitoreo en tarjetas Raspberry Pi modelo 3 B+ con el módulo de expansión GrovePi+.
- Sintonizar e implementar un controlador PID utilizando el método experimental de Ziegler-Nichols.

1.9. Metodología

El proyecto se lleva a cabo por medio de etapas, cada una de ellas describe a continuación.

- Etapa 1. Diseño del invernadero: El invernadero cuenta con dimensiones de 1mx 0.45mx 0.728m a base de madera de 0.02m de grueso. Tubería PVC para la estructura, codos, derivaciones de 3 y 4 vías con el fin de interconectar cada tubo, así como también lona calibre 800 para forrar el exterior del invernadero. La fijación de los tubos y conectores se lleva a cabo con pegamento para PVC y en el caso de los 4 tubos que sostienen la estructura del invernadero, serán fijados a la base de madera por medio de tornillos. Se utilizan placas de madera para dividir el invernadero en dos niveles en donde, el primero es un espacio destinado para las plantas, la instalación de sensores y el bombillo cerámico emisor de calor. El segundo nivel es para el resguardo de los elementos del sistema de monitoreo, esto es, Raspberry Pi, GrovePi+, circuitos electrónicos, fuentes de alimentación, extensión multi contacto para conexión a la red de voltaje de 120V AC, la serie de luz para crecimiento de plantas y la pantalla display para la visualización de interfaces.

- Sistema electrónico para el controlador PID: Diseño y elaboración de PCB de los circuitos electrónicos por medio de *Proteus* y *LTspice*. Instalación de los circuitos electrónicos que interconectan a ambas tarjetas de control con el bombillo cerámico emisor de calor para el controlador PID. El primer circuito tiene como principal componente el circuito integrado H11AA2X mientras que el segundo lleva consigo un amplificador operacional TL084CN y un optoacoplador MOC3021, así como un Triac BT139 a 16A. El *hat* GrovePi+ que lleva consigo el microcontrolador ATMEGA328P-AU se encuentra montada en los pines de la Raspberry Pi 3 B+, en ellas los sensores y los circuitos electrónicos están conectados por medio de cables universales de 4 vías de la marca Grove. Estos circuitos al igual que las tarjetas estarán unificados en una base de triplay mientras que las fuentes de alimentación y el multi contacto permanecerán a un lado de ellas.

- Etapa 3. Desarrollo del software: Se lleva a cabo la instalación de librerías necesarias, para que posteriormente se desarrollen los scripts necesarios para las interfaces gráficas de usuario que conforman al sistema de monitoreo. El entorno de programación utilizado es Thonny IDE (instalado de forma nativa en el sistema operativo de la Raspberry Pi). El control PID y el filtro de Kalman, así como las interfaces son desarrolladas mediante el uso de Python. Se realiza la configuración de los puertos de la GrovePi+ para definir las entradas y salidas de los componentes externos (circuitos electrónicos y sensores) así como el control de pulso PWM. Es en esta etapa que también se obtiene la curva de reacción de la temperatura del invernadero y el programa necesario para mostrar gráficamente la tangente que sirve como guía para encontrar los parámetros necesarios de cada ganancia del controlador PID.

- Etapa 4. Funcionamiento de los circuitos electrónicos: Ambos circuitos electrónicos serán verificados con el osciloscopio, esto es, validar las señales de entrada y salida (amplitud, periodo y frecuencia). En el caso del primer circuito, se validará la entrada de voltaje de la red de 120V AC (senoide) y a la salida los pulsos resultantes de 4.7V DC a la misma frecuencia de la señal de entrada. Para el segundo circuito, se validará la señal de entrada la cual proviene de la Raspberry Pi (pulso cuadrado de 3.44V DC), su amplificación a la salida del circuito TL084CN y la salida del Triac BT139 actuando en conjunto con el optoacoplador MOC3021 para regular lo más que se pueda la energía del actuador térmico.

- Etapa 5. Funcionamiento de las interfaces gráficas: Pruebas de funcionamiento de las interfaces, desde la ventana principal que muestra las lecturas de cada uno de los sensores y el menú de funciones, hasta el control PID mediante la interfaz. Se verificará que cada una de las funciones del menú logre ejecutarse correctamente. La ventana de los gráficos de temperatura, luz y humedad de suelo se corrobora con la conexión de los sensores y por observación que las líneas 2D se muestren en cada plot, mientras que la interfaz del controlador requiere de los dos circuitos electrónicos. Para validar que el programa del controlador sea correcto se tienen como indicadores las etiquetas de la ventana, en ellos se arrojan datos de error, suma del error, valor del PID y el valor del pulso PWM. Por último, para la interfaz de grabado de datos se verificará el programa por medio de los archivos generados en Excel, al abrirlos deberá contener información del tiempo en que se toma la muestra de temperatura y el valor captado por el sensor.

- Etapa 6. Integración de todos los elementos: Se conectarán e integrarán por completo todos los elementos para conformar el sistema de monitoreo. Se fijará el bombillo cerámico a las placas de madera que dividen el interior del invernadero, en el segundo nivel se organizaran los elementos de control del sistema y se ejecutara desde el entorno de desarrollo Thonny IDE el software el monitoreo de variables físicas y el control de temperatura.

Capítulo 2

Marco Teórico

Durante este capítulo se describen los conceptos fundamentales para llevar a cabo el sistema de monitoreo, el controlador PID, la generación de los circuitos electrónicos y llegar a obtener resultados satisfactorios. Conceptos desde sistemas de control, sus clasificaciones, el filtro de Kalman, la metodología para la sintonización de ganancias del controlador, requerimientos para desarrollar e implementar el sistema de monitoreo con Python, la descripción de las tarjetas de desarrollo, sus funcionalidades, entre otros.

2.1. Sistema de control

Búsqueda de medios, métodos o formas con el propósito de alcanzar un objetivo, estos sistemas además de diseñarse para cumplir ciertas tareas, requieren de estrategias de control que logren hacerlo posible [4]. En la industria los sistemas de control son ampliamente utilizados para realizar tareas de manera rutinaria, como por ejemplo las líneas de ensamble automatizadas, la robótica, el control de maquinaria, etc. Los sistemas de control por lo general están estructurados de la siguiente manera:

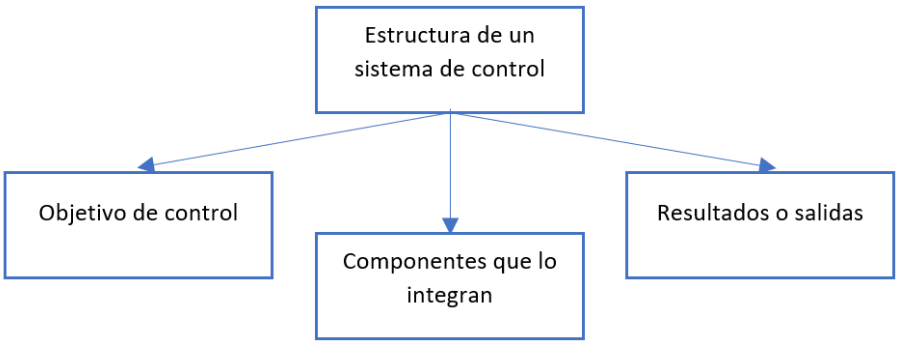


Figura 2.1: Estructura de un sistema de control.

El objetivo de un sistema de control es manipular la salida de un variable controlada a partir de la entrada por medio de los elementos que integran al sistema de control.

2.1.1. Sistema de control de lazo abierto

En los sistemas de lazo abierto, no se requiere una realimentación para alcanzar un valor deseado en la salida del proceso [2]. Utiliza un controlador y un actuador para obtener una respuesta deseada sin necesitar de una realimentación tal como se muestra en la Figura 2.2.

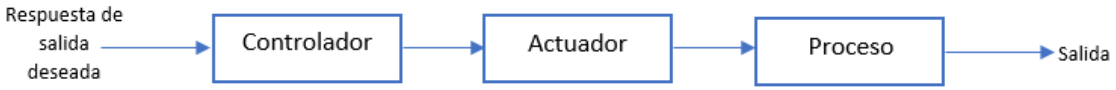


Figura 2.2: Sistema de control de lazo abierto [2].

2.1.2. Sistema de control de lazo cerrado.

En un sistema de control de lazo cerrado, se puede encontrar una señal de realimentación proveniente de la salida que tiene como propósito comparar la salida real del proceso con la respuesta de la salida deseada. La diferencia entre estas dos es utilizada como medio de control de tal manera que el error se reduzca continuamente [4]. La señal de realimentación es fundamental ya que conforma el análisis y

el diseño de los sistemas de control.

La manera de representar un sistema de control en lazo cerrado se presenta en la Figura 2.3, en la cual se puede visualizar de mejor manera la señal de realimentación. El gráfico muestra el papel del sensor como un elemento que cuantifica el valor de salida actual para obtener el error.

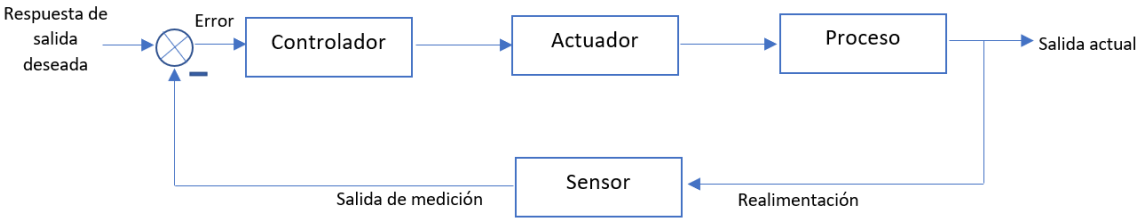


Figura 2.3: Sistema de control de lazo cerrado [2].

2.1.3. Sistemas de control en tiempo continuo

Un sistema de control en tiempo continuo todas las partes de un sistema son variables en función del tiempo t, es decir, conocidas en cualquier instante de tiempo [42]. En los sistemas de control de tiempo continuo las señales se clasifican de CA y CD [4].

En un sistema de control CA las señales están moduladas, caso contrario a los de CD que no lo están. Como componentes comunes en los sistemas CD se encuentran los potenciómetros, amplificadores CD, motores de CD, tacómetros CD, por mencionar algunos. En el caso de los sistemas de control en CA los componentes comunes son los giroscopios, acelerómetros, motores de CA, entre otros.

Otra de las características de estos sistemas es la descripción de las señales por medio de ecuaciones diferenciales.

2.1.4. Sistemas de control en tiempo discreto

Un sistema de control en tiempo discreto a diferencia de uno continuo tiene la característica en la cual las señales en uno o mas puntos del sistema se encuentran como pulsos o bien como código digital [4]. Los sistemas de tiempo discreto se dividen a su vez en sistemas de control de datos muestreados y sistemas de control digital.

En un sistema de control de datos muestreados desde un punto de vista general, utiliza señales en forma de pulsos de datos, en contraparte un sistema de control digital requiere del uso de una computadora o un controlador digital en el sistema, de allí que la forma de las señales se encuentren en código digital (o también conocido como código binario). En la Figura 2.4 se muestra el diagrama de bloques caracte-

rístico en un sistema de control digital. En dicha ilustración se presentan elementos tales como:

- Un muestreador y retenedor (S/H). Es un circuito que recibe como entrada una señal analógica mantenida de manera constante un determinado tiempo
- Convertidor analógico-digital (A/D). Cambien conocido como codificador, es un dispositivo que convierte una señal analógica en una digital de manera numérica.
- Convertidor digital-analógico (D/A). Cambien conocido como decodificador, es un circuito que convierte una señal digital en una señal analógica, puede verse como una interfaz entre un componente digital y uno analógico.
- Planta o proceso. Objeto a controlar, como por ejemplo un horno, un reactor químico o un sistema de seguimiento.

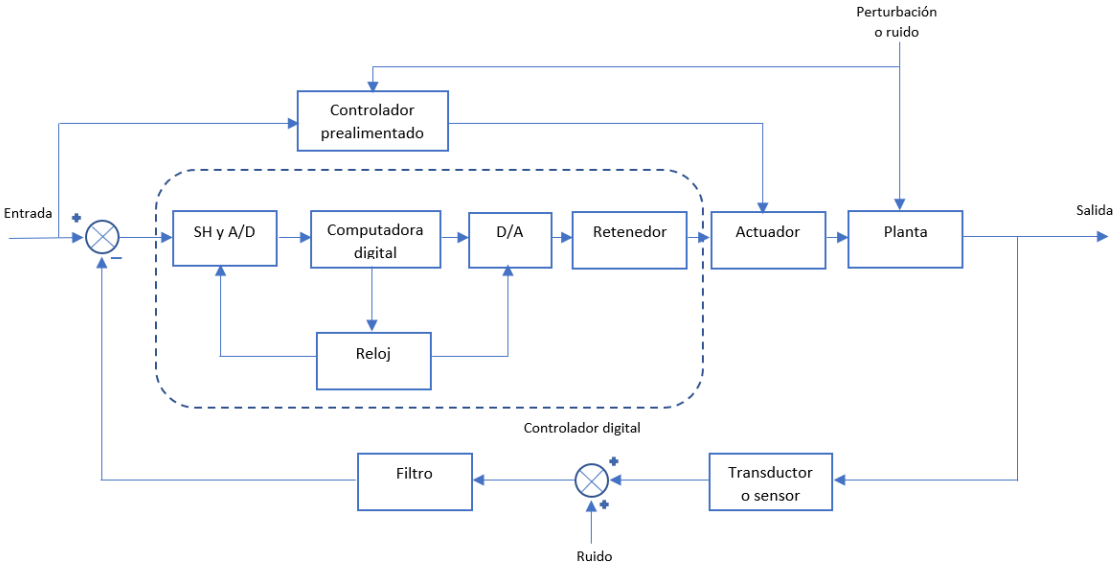


Figura 2.4: Diagrama de bloques de un sistema de control digital [3].

En el sistema de control digital de la Figura 2.4 se puede observar un controlador realimentado el cual funge como un método de control adicional utilizado para corregir desviaciones del sistema.

Ademas de la planta, es posible encontrar ruido en la señal del tranductor o del sensor utilizado para medir la salida por lo que el uso de filtros es necesario para mitigar componentes indeseadas.

2.1.5. Ventajas del control digital.

El control digital ofrece ventajas por sobre el analógico, principalmente se encuentran las siguientes:

- Los componentes digitales son menos susceptibles al envejecimiento así como a las variaciones en las condiciones ambientales.
- Los procesadores digitales miden y pesan menos, son versátiles y ofrecen una amplia gama de aplicaciones enfocadas al control.
- Los microprocesadores son mas flexibles con la programación, un cambio en el diseño se realiza con software.
- Confiabilidad.

2.1.6. Controlador automático

Un controlador automático lleva a cabo una comparación del valor real de la salida de una planta con la entrada de referencia. Se necesita determinar la desviación y en base a ello se produce una señal de control con el objetivo de reducir esa desviación a cero o bien aproximarlo a un valor más pequeño. Para ello se requiere de una acción de control, el cual se define como la manera en que el controlador automático produce una señal de control. Dicha acción puede desprender varios tipos los cuales se diferencian por el modo en que operan en una planta. En la Figura 2.5 se representa gráficamente los componentes que integran un proceso controlado.

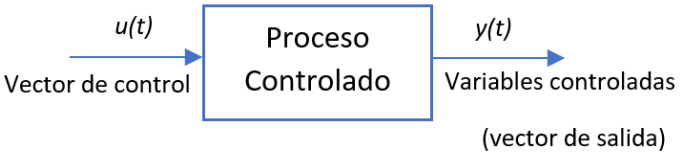


Figura 2.5: Proceso controlado [4].

Las principales tareas de un controlador generalmente abarcan 3 puntos clave; el que deberá hacer el sistema, las configuraciones o bien la selección del controlador para determinar la conexión con el proceso a controlar y la obtención de los parámetros del control.

2.1.7. Clasificación de controladores industriales.

La manera en que se clasifican los controladores industriales radica en la acción de control que adopta cada uno de ellos, se pueden encuentran los siguientes:

- De dos posiciones o controladores ON-OFF.
- Controladores proporcionales.
- Controladores integrales.
- Controladores proporcionales-integrales.
- Controladores proporcionales-derivativos.
- Controladores proporcionales-integrales-derivativos.

2.1.8. Controlador proporcional integral derivativo (PID).

El controlador PID es considerado una herramienta/instrumento utilizado en sistemas de control de procesos de tipo industrial. Las siglas PID hacen referencia a proporcional, integral y derivativo, acciones de control que influyen en la toma de decisiones tomando en cuenta distintos factores. El algoritmo de este controlador está definido por la ecuacion 2.1.

$$u(t) = K\left(e(t) + \frac{1}{T_i}\int_0^t e(\tau)\,d\tau + T_d\frac{de(t)}{dt}\right) \tag{2.1}$$

Donde, u es la variable de control y e es el error del control. Tal como se aprecia en la ecuación 2.1, la variable de control está construida por 3 términos; P (proporcional al error), I (proporcional a la integral del error) y D (proporcional a la derivada del error).

El control PID consiste en un algoritmo capaz de implementarse para tener los sistemas funcionando correctamente, dicho algoritmo puede interpretarse y llevarse a la práctica de distintas maneras. Uno de los términos claves es el control de bucle cerrado, es decir, que el sistema este constantemente monitoreando su salida para posteriormente realizar un ajuste.

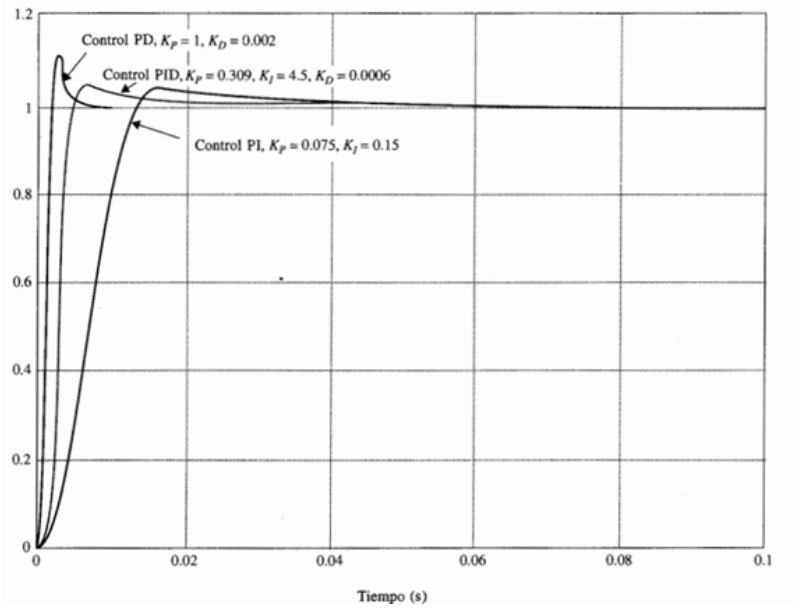
Tiene la característica de implementarse digitalmente por medio de la programación del algoritmo desde una computadora con el uso de distintos lenguajes de alto nivel hoy en día.

En todo control PID se requiere de una forma de determinar los parámetros con los que funcionara (obtención de ganancias), la sintonía del controlador es el proceso de selección de dichos parámetros que cumplan con las especificaciones de comportamiento dadas por la planta.

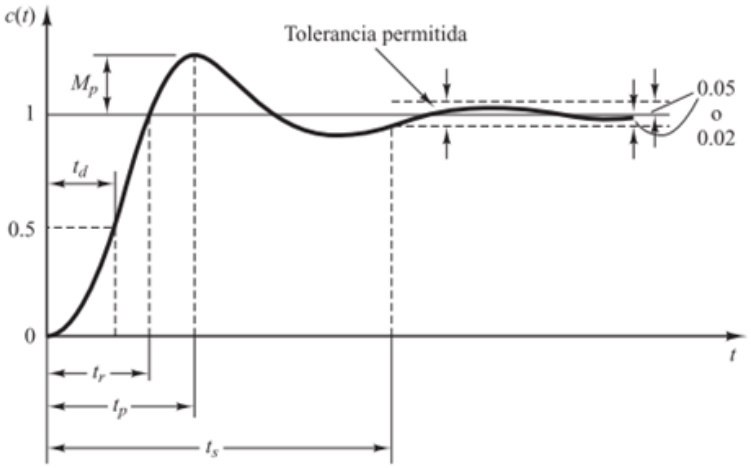
En la práctica, las características de desempeño requeridas en un sistema de control se especifican en términos de cantidades en el dominio de tiempo [26]. Mas en concreto, en términos de la respuesta transitoria para una entrada escalón unitario capaz de mostrar oscilaciones amortiguadas antes de alcanzar el estado estacionario, como se puede observar en la Figura 2.6b. Para ello se requieren de algunos parámetros tales como:

- Tiempo de retardo (t_d). Tiempo que le toma al sistema llegar al 50 % del valor final.
- Tiempo de subida (t_r). Tiempo requerido para que la respuesta del 0 % al 100 %.
- Tiempo pico (t_p). Tiempo requerido para que la respuesta alcance el primer pico de sobreelongación.

- Sobreelongación (M_p). El máximo sobreimpulso es el máximo valor del pico de la curva de respuesta
- Tiempo de asentamiento (t_s). Tiempo que se requiere para que la curva de respuesta alcance un rango cercano del valor final por lo general de 2 % o 5 %.



(a) Respuestas al escalón de un sistema con los controladores pi, pd y pid.



(b) Curva de respuesta a escalón unitario.

Figura 2.6: Respuesta del control PID [4].

2.1.9. Método de sincronización de Ziegler-Nichols.

En este método, la respuesta de la planta a una entrada escalón unitario se obtiene de manera experimental, por lo que se debe mostrar una curva tal y como se muestra en la Figura 2.7.

La curva con forma de S se caracteriza por dos parámetros; El tiempo de retardo **L** y la constante de tiempo **T** (que en este trabajo se le denominara τ), en donde, para encontrar su valor es necesario dibujar una recta tangente en el punto de inflexión de la curva y determinar las intersecciones de la tangente con el eje de tiempo y con la linea $c(t) = K$ de acuerdo con la Figura 2.7 [26].

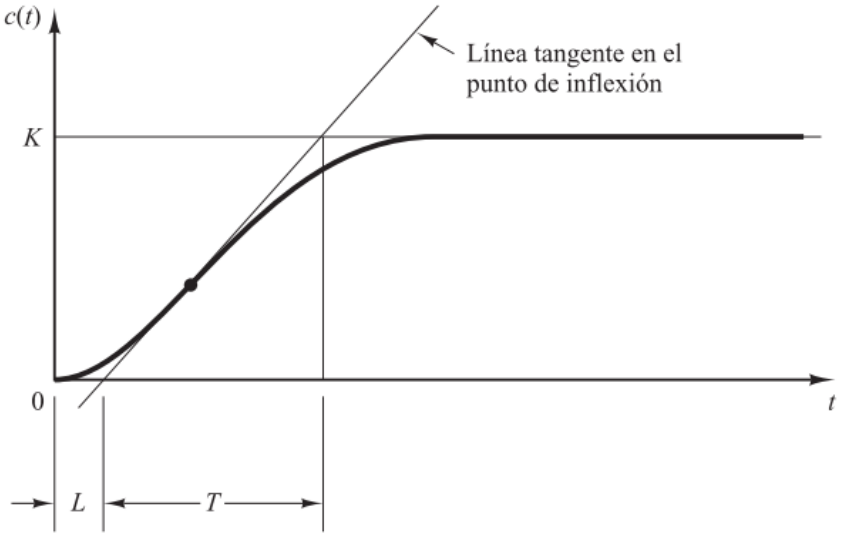


Figura 2.7: Curva de respuesta del método 1 de Ziegler-Nichols.

Definida la curva de respuesta, se emplea una serie de reglas muy útiles para determinar los valores K_p, K_i y K_d propios de un controlador proporcional-integral-derivativo. Estas reglas resultan ser adecuadas para el caso de no saber el modelo matemático de la planta.

En la Tabla 2.1 se presentan las reglas de sintonía de Ziegler y Nichols para la obtención de ganancias según el tipo de controlador a implementar, desde el control proporcional, proporcional-integral hasta el proporcional-integral-derivativo. Se obtiene un ajuste óptimo con estas fórmulas con el objetivo de minimizar la integral del término de error (área mínima de error). El tiempo de retardo efectivo (L) será el efecto principal y el componente del valor del tiempo integral. La pendiente, o también conocida como tasa de cambio del proceso, será el factor principal que influye en el ajuste de la ganancia del controlador, KQ, ya que representa la ganancia o sensibilidad del proceso en sí [43].

Tipo de controlador	$\mathbf{K_p}$	$\mathbf{T_i}$	$\mathbf{T_d}$
P	$\frac{T}{L}$	∞	0
PI	$0{,}9\frac{T}{L}$	$\frac{L}{0{,}3}$	0
PID	$1{,}2\frac{T}{L}$	$2L$	$0{,}5L$

Tabla 2.1: Regla de sintonía de Ziegler Nichols basada en la respuesta escalón de la planta [26].

2.2. Filtros.

Un filtro de acuerdo con F. Miyara [44] se define como un dispositivo que modifica de algún modo una señal que pasa a través de el. Dependiendo del interés practico, existen distintos tipos de filtros como por ejemplo: los filtros eléctricos, los filtros mecánicos, filtros acústicos, filtros ópticos, por mencionar algunos. También pueden clasificarse de acuerdo al comportamiento y su modelo matemático como lineales o no lineales, analógicos o digitales.

Un filtro analógico es aquel en el cual una señal puede tomar cualquier valor dentro de un intervalo (continuo), mientras que en un filtro digital la señal toma valores discretos (muestreado). Los filtros digitales tienen aplicaciones en procesos de carácter industrial, en sistemas de control y en el monitoreo [45] en plantas químicas, sistemas de control de vuelo, en sistemas de adquisición de datos, en ramas de la medicina, entre otras aplicaciones. Esta clase de filtro también es conocido como Filtros Digitales en Tiempo Real (FDTR). Un FDTR se implementa en microcontroladores e inclusive en computadoras con sistema operativo en tiempo real (conocidos como SOTR y que son utilizados para cumplir con operaciones en tiempos muy específicos).

F. Miyara [45] describe que un filtro digital se aplica para realizar alguna de las siguientes tareas:

- Eliminar el ruido de los datos de un sistema.
- Extraer información de acurdo a alguna característica del sistema.
- Predecir el comportamiento del sistema a analizar.
- Reconstruir el comportamiento del sistema a analizar.

El proceso de filtrado esta basado en acciones como el monitoreo de estados emitidos por el sistema a analizar y una etapa de predicción o identificación del ruidos que son originados por el sistema, por un error de lectura del receptor o por influencia del medio ambiente que rodea al emisor y al receptor.

Los filtros digitales brindan un mejor rendimiento, flexibilidad ademas de tener una baja sensibilidad a las variaciones paramétricas si se compara con los filtros analógicos.

2.2.1. Importancia de los filtros en los sistemas de control.

De acuerdo con G. Ellis [46] los filtros se encuentran en el controlador del sistema de control, en los dispositivos de retroalimentación en inclusive en la planta. En un controlador son requeridos para eliminar ruido, reducir el Aliasing y enfrentar problemas con la resonancia. En ocasiones los filtros poseen características fijas o bien dan la posibilidad de ajustarse haciéndolos mas flexibles para ejecutar con éxito el sistema de control.

Los filtros presentes en lazos de realimentación tienen la particularidad tal que, para ciertos procesos físicos actúan como filtros, esto se ve por ejemplo en los sensores de temperatura como en termistores los cuales poseen inercia térmica. Cuando se presenta un cambio en la temperatura, la masa térmica del sensor tarda un tiempo en cambiar. De manera que una gran variedad de sensores eléctricos y electrónicos tienen una etapa de amplificación activa para convertir señales de bajo voltaje a niveles que puedan ser transmitidos al controlador. La amplificación también esta sujeta a ruido.

Los filtros adaptativos por su parte representan una parte importante del procesamiento estadístico de señales ya que históricamente su enfoque paramétrico a sido el principal enfoque de la ingeniería en el procesamiento de señales [47]. El enfoque no paramétrico es el que se basa en utilizar modelos mas generales para replicar el comportamiento deseado con el uso de información estadística de un conjunto de datos representativos. Los filtros adaptativos se basan en un enfoque intermedio entre ambos, han demostrado mejorar en comparación a los filtros convencionales. Los campos en los que han repercutido van desde en las comunicaciones , la ingeniería de control, en robótica, en la ingeniería biomédica por mencionar algunos de los mas importantes.

2.2.2. Aplicaciones de los filtros.

Un filtro digital es empleado para alguna de las siguientes acciones [48]:

- Acondicionamiento de la señal de entrada. Con el fin de eliminar componentes ruidosas en la señal captada por los sensores, en la selección y eliminación de frecuencias de interés.

- Eliminación de frecuencias no deseadas. Evitar problemas de Aliasing (aparición de ondas con diferente frecuencia) así como para determinar el ancho de banda para compresión o transmisión.

- Acondicionamiento de señal producida. Para la eliminación de armónicos, en la supresión de interferencias conducidas y radiadas y para la mejora del rendimiento.

A continuación se presentan los principales filtros tanto analógicos como digitales.

Filtros analógicos.

Los filtros analógicos se dividen en dos categorías, pasivos y activos. En donde, los filtros pasivos están conformados por resistencias, condensadores e inductores. Mientras que en los filtros activos se requiere el uso de amplificadores operacionales.

- Filtro Butterworth. Diseñado para obtener una respuesta con mínimas ondulaciones (plana) con respecto a la frecuencia de corte, dicha salida se mantiene constante aproximadamente a la frecuencia de corte.

- Filtro Chebyshev. Filtro electrónico con la capacidad de diseñarse en digital, con este tipo de filtro se obtiene una caída de la respuesta en frecuencia mas larga en frecuencias bajas debido al rizado de las bandas (paso rechazo).

- Filtro Bessel. Se tratan de filtros utilizados en aplicaciones de audio debido a su linealidad, característica que evita la distorsión de señales ademas de tener una mayor zona de transición entre las bandas pasantes y no pasantes.

Filtros digitales.

- Filtro FIR. Este tipo de filtro esta en la categoría de los no recursivos debido a la ausencia de realimentación y cuya respuesta es finita al impulso. Al decir que es de respuesta finita si en la entrada se encuentra en cero por n periodos de forma consecutiva a la salida tendrá el mismo valor.

- Filtro IIR. A diferencia del filtro FIR un filtro IIR es de respuesta infinita al impulso, mantiene realimentación y en consecuencia es de tipo recursivo. La salida del filtro depende de la entrada actual así como de las salidas anteriores utilizadas por medio de una ecuación en diferencias.

- Filtro de media móvil. Es uno de los filtros mas comunes dada su facilidad de uso. Capaz de reducir el ruido aleatorio, sin embargo, tiene muy poca capacidad para separar una banda de frecuencias de otra.

- Filtro de mediana. Es un filtro que funge como una técnica de mejoramiento de señal en el procesamiento digital de imágenes. Pese a su utilidad presenta alto coste computacional al realizar un procesamiento lento.

- Filtro de Kalman. Desarrollado por Rudolf E. Kalman en 1960 es útil para identificar el estado no visible de sistemas dinámicos, caso parecido al observador de Luenberger. Es una buena alternativa cuando el sistema esta sometido a ruido blanco y se conocen los ruidos que afectan al sistema.

- Filtro de Wiener. Este filtro trabaja en base con métodos estadísticos, logrando reducir el ruido de la señal de entrada provocando que la señal de salida sea lo mas cercano a la señal deseada.

2.2.3. Ruido de medición

El ruido de medición es todo aquel error aleatorio o fluctuación que se llega a producir al realizar una medida de una cantidad física. Entre las características del ruido se presentan las siguientes:

- Naturaleza: Se modela como una variable aleatoria de media cero, es decir que su valor promedio es cero. Provoca estimaciones imparciales del valor verdadero que se está midiendo.
- Covarianza: El ruido de medición se caracteriza por su covarianza, la cual cuantifica que tanto varía el ruido.
- Independencia: El ruido en un momento no llega a afectar el ruido en otro momento, de manera que se puede logra el análisis y diseño de algoritmos de estimación como por ejemplo el filtro de Kalman.
- Impacto en la estimación: A menudo que aumenta el ruido de medición también aumenta la incertidumbre en las estimaciones.

Una estimación estatal representa la mejor conjetura de la condición interna de un sistema en un momento dado, esencial para controlar un sistema de manera efectiva.

Ruido blanco

Es un tipo de señal aleatoria de igual intensidad en diferentes frecuencias. Fundamental en el procesamiento de señales y datos estadísticos. Una de sus características es su densidad espectral de potencia constante, esto quiere decir que contiene todas las frecuencias a niveles de potencia iguales. Este tipo de señal puede encontrarse en tiempo continuo y en tiempo discreto:

- Ruido blanco en tiempo continuo: En sistemas de tiempo continuo el ruido blanco tiene correlación infinita consigo mismo en el momento actual y correlación cero en cualquier intervalo de tiempo distinto de cero.
- Ruido blanco en tiempo discreto: En sistemas de tiempo discreto no se correlaciona consigo mismo en diferentes puntos de tiempo.

2.3. Filtro de Kalman.

El filtro de Kalman es un algoritmo matemático generalmente utilizado para la estimación de estados de un sistema dinámico en base a una serie de mediciones con presencia de ruido a lo largo del tiempo. Es utilizado para la solución de problemas de estimación [6].

Por “estado” se entiende como la condición o la posición que se está observando. El algoritmo proporciona dos tipos de salidas, una de ellas es la covarianza conocida

como la incertidumbre de la estimación y la otra que es la media del estado entendida como la estimación del filtro. Diseñado para ser optimo bajo ciertas condiciones, cuando el proceso y las mediciones ruidosas son gaussianos, de media cero y no correlacionados.

El filtro de Kalman trabaja basándose en 2 etapas, la primera de predicción y otra de corrección englobadas en las expresiones matemáticas descritas por Simon (2006).

2.3.1. La necesidad del filtro de Kalman en el control de procesos térmicos.

Cuando se emplea un controlador PID, la precisión de la señal de realimentación (en este caso, la temperatura medida) es crítica, ya que la acción de control se basa en el cálculo de tres términos sensibles al ruido: proporcional, integral y derivativo. La presencia de ruido puede producir acciones de control erráticas debido a cambios abruptos en la medición, acumulación errónea en el término integral, lo que genera *offset* o sobreimpulsos, amplificación del ruido por el término derivativo, que reacciona fuertemente a pequeñas variaciones, por mencionar algunos problemas.

El filtro de Kalman se plantea como una herramienta computacionalmente eficiente que permite estimar el estado real del sistema (la temperatura verdadera). Debido a su naturaleza, reduce el impacto del ruido y por tanto, mejorar la precisión de la señal usada por el controlador PID, permitiendo así el desempeño del controlador.

Por ejemplo, en el control de un horno la lectura de la temperatura puede fluctuar rápidamente por el ruido eléctrico. Si el PID actúa sobre estas señales sin filtrar tenderá a sobreajustar la señal de control, esto a suvez causa oscilaciones en el calentador. Al aplicar un filtro de Kalman, se obtiene una estimación más estable y precisa de la temperatura, lo que permite al controlador operar de manera más eficiente, evitando sobreimpulsos, mejorando el tiempo de establecimiento y reduciendo el error en estado estacionario.

En el año 2018, se propuso un controlador PID apoyado con un filtro de Kalman dirigido hacia un invernadero. El cual mejoro en gran medida el rendimiento y la respuesta del sistema , mas en especifico en aspectos clave como la convergencia rápida en un tiempo de 0.2 segundos demostrando una respuesta superior a comparación de los métodos convencionales PID que mostraron tener problemas para estabilizar su salida [40]. Logro reducir las fluctuaciones ya que en un valor de referencia de 25° C con el filtro de Kalman la respuesta oscilo entre 24° C y 26° C mientras que sin el, el rango se ampliaba de 22° C a 30° C.

Por tanto, se justifica la incorporación del filtro de Kalman en el sistema de control para garantizar una retroalimentación más limpia, que mejore la estabilidad y el rendimiento del sistema de control digital.

Algoritmo del filtro de Kalman.

En la Figura 2.8 se puede observar de manera gráfica el algoritmo, mostrando su forma recursiva en base con el cálculo de predicción y corrección del estado dependiendo de la medición obtenida (y_k).

El algoritmo lleva los siguientes pasos.

Primero se necesita de dos estados iniciales denotados como $\hat{x}_0^+$ y P_0^+:

- $\hat{x}_0^+$ = Estimación inicial de x_0 antes de conocer las mediciones, si no se tienen las mediciones se debe plantear el valor esperado del estado inicial de x_0.
- P_0^+ = Covarianza de la estimación inicial $\hat{x}_0^+$, representa la incertidumbre de la estimación inicial x_0.

El segundo paso, consta de predecir el estado del sistema en el paso siguiente (Ecuación 2.2) así como la incertidumbre de la predicción (Ecuación 2.3).
El tercer paso es computarizar la ganancia de Kalman incluyendo la matriz de covarianza del ruido de los sensores (Ecuación 2.4).
En el cuarto paso, al recibir la medición (y_k) se procede a "corregir" la predicción previa (Ecuación 2.5) y posteriormente la incertidumbre del estado actual (Ecuación 2.6).

Expresiones matriciales para el calculo de la etapa de predicción y corrección.

Etapa de predicción.

$$\hat{x}_k^- = F_{k-1}\hat{x}_{k-1}^+ + G_{k-1}u_{k-1} \tag{2.2}$$

$$P_k^- = F_{k-1}P_{k-1}^+F_{k-1}^T + Q_{k-1} \tag{2.3}$$

En donde, las Ecuaciones 2.2 y 2.3 representan la predicción del estado. En ellas se encuentran las matrices F y G las cuales representan:

- F. Matriz de transición del estado (tamaño $n_x \times n_x$).
- G. Matriz de control (tamaño $n_x \times n_u$).
- u_{k-1}. Variable de entrada que es medible (determinista, de tamaño $n_u \times 1$).
- $\hat{x}_{k-1}^+$. Vector de estado del sistema estimado en el instante k-1.
- $\hat{x}_k^-$. Vector de estado del sistema predicho en el instante k.
- P_k^-. Predicción de la incertidumbre en el instante k.
- P_{k-1}^+. Incertidumbre del estado estimado en el instante k-1.

- Q_{k-1}. Matriz de covarianza del ruido de proceso.

Etapa de corrección.

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} \tag{2.4}$$

$$\hat{x}_k^+ = \hat{x}_k^- + K_k(y_k - H_k \hat{x}_k^-) \tag{2.5}$$

$$P_k^+ = (1 - K_k H_k) P_k^- \tag{2.6}$$

En donde, la Ecuación 2.4 representa el calculo de la ganancia optima o bien la ganancia de Kalman, se puede observar que tiene presente la variable H y R, las cuales representan:

- H. Matriz de observación, representa las variables que se están midiendo.
- R. Matriz de covarianza del ruido del sensor.
- K_k. Ganancia de Kalman.
- P_k^-. Actualización de la matriz de covarianza.

Por otra parte, en la Ecuación 2.5 se presenta la ecuación de corrección del estado, en la cual ya interviene la ganancia de Kalman optima (K_k), la matriz de Observación (H) y la lectura de entrada del sensor representada por y_k. Estas matrices son necesarias para completar la fase de corrección, en el caso de la Ecuación 2.6 representa la corrección de la covarianza.

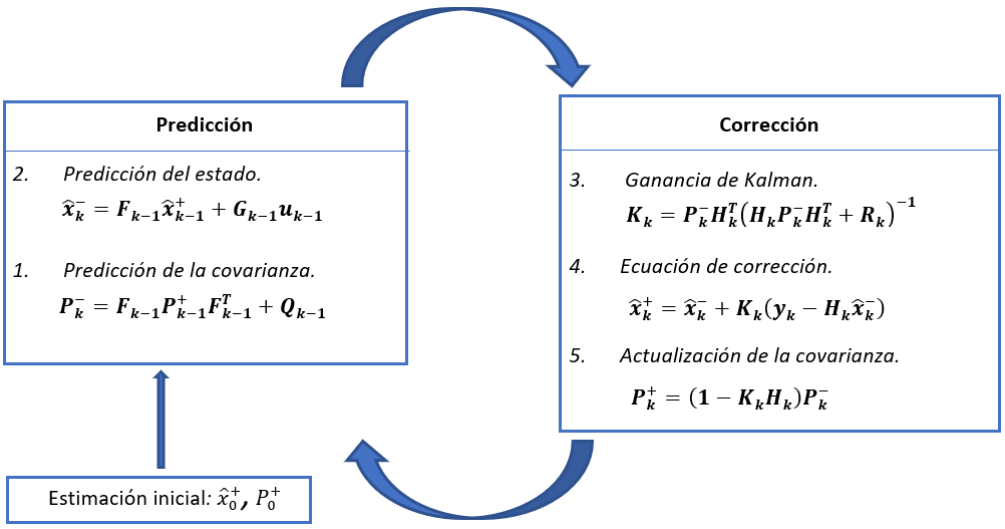


Figura 2.8: Algoritmo del filtro de Kalman.

Covarianza

La covarianza en el filtro de Kalman es descrita como la incertidumbre o bien la dispersión de las estimaciones del estado alrededor de la media. Por medio de la covarianza se cuantifica la seguridad del filtro con respecto a su estimación, esto es,

cuanto mayor sea la covarianza mayor es la incertidumbre sobre la estimación del estado y mayor influencia tendrá sobre las nuevas mediciones causando un mayor impacto en la actualización. Es por ello que es preferible una menor covarianza ya que se tendrá una mayor confianza en la estimación previa y las mediciones futuras tendrán un menor impacto.

Otros de los términos relevantes es la media y el valor esperado, por lo general la media representada por μ y el valor esperado por E. En donde, la media de una serie de valores se denota por la ecuación:

$$\mu = V_{mean} = \frac{1}{N}\sum_{n=1}^{N} V_n \tag{2.7}$$

El filtro de Kalman depende de la varianza del ruido de proceso y de la varianza del ruido medido, al obtener el valor promedio de una tira de datos es posible obtener la varianza usando la expresión matemática mostrada a continuación:

$$\sigma^2 = \frac{1}{N}\sum_{n=1}^{N}(x_n - \mu)^2 \tag{2.8}$$

2.3.2. Comparación de filtros digitales con respecto al filtro de Kalman

A continuación se presenta por medio de las Tablas 2.2 y 2.3, una comparativa de los principales filtros digitales, incluyendo sus características, ventajas, desventajas y aplicaciones comunes.

En cada Tabla se encuentran tanto filtros recursivos como no recursivos y sus parámetros clave como la complejidad computacional siendo uno de los aspectos mas relevantes a la hora de implementarse digitalmente, su estabilidad en general, la adaptabilidad, así como el grado de precisión acorde a los campos de aplicación de cada uno de ellos.

Esto con el objetivo de identificar cual es mas indicado en distintos escenarios, destacando principalmente el filtro de Kalman como la mejor opción en entornos dinámicos con presencia de ruido comúnmente encontrados en aplicaciones de control y automatización.

Criterio	**FIR**	**IIR**	**Media móvil**	**Mediana**
Tipo	No recursivo	Recursivo	No recursivo	No recursivo
Definición	Usa una suma ponderada de muestras pasadas de la entrada.	Usa retroalimentación, la salida depende de entradas y salidas previas.	Promedia un número fijo de muestras recientes.	Sustituye cada valor por la mediana de un conjunto de valores cercanos.
Complejidad computacional	Alta para filtros de alta precisión.	Baja, usa menos coeficientes.	Muy baja, solo suma y divide.	Baja, requiere ordenamiento.
Consumo de memoria	Alto, almacena varias muestras previas.	Bajo, almacena pocas muestras previas.	Bajo, almacena solo N muestras.	Bajo, almacena solo N muestras.
Estabilidad	Siempre estable.	Puede ser inestable si está mal diseñado.	Siempre estable.	estable. Siempre estable.
Tipo de respuesta	Lento para transiciones rápidas.	Rápido, se ajusta rápidamente a cambios.	Lento para cambios súbitos.	Puede ser lento en transiciones.
Capacidad de adaptación	Baja, coeficientes fijos.	Baja, depende del diseño.	Baja, depende del tamaño de la ventana.	Baja, depende del tamaño de la ventana.
Precisión en entornos ruidosos	Moderada, no optimiza ruido.	Baja si el ruido es fuerte.	Baja, no diferencia señal de ruido.	Buena si el ruido es impulsivo.
Casos de aplicación	Procesamiento de audio, imagen, video, comunicaciones digitales.	Procesamiento de voz, control en tiempo real, electrónica de potencia.	Sensores de temperatura, indicadores financieros, filtrado de datos en tiempo real.	Procesamiento de imágenes, eliminación de ruido en sensores.
Ejemplos	Filtro FIR en ecualizadores de audio.	Filtro IIR en procesamiento de voz.	Media móvil en indicadores de bolsa.	Mediana en reducción de ruido de imágenes.

Tabla 2.2: Comparación entre distintos tipos de filtros digitales parte 1

Criterio	**Wiener**	**Kalman**	**EWMA**
Tipo	Recursivo	Recursivo	Recursivo
Definición	Minimiza ruido basándose en la estimación de la señal y ruido.	Estima el estado de un sistema dinámico minimizando el error.	Promedia exponencialmente los datos previos.
Complejidad computacional	Moderada, necesita conocer características del ruido.	Alta, requiere modelos dinámicos.	Baja, simple actualización recursiva.
Consumo de memoria	Moderado, almacena información de ruido y señal.	Moderado, almacena estados previos.	Bajo, solo mantiene una media actualizada.
Estabilidad	Estable si se configura bien.	Estable, optimizado para minimizar errores.	Siempre estable.
Tipo de respuesta	Rápido si el ruido es bien modelado.	Rápido y óptimo.	Rápido pero con retardo.
Capacidad de adaptación	Media, depende de ruido estimado.	Alta, se ajusta dinámicamente.	Media, ajustable con el factor de suavizado.
Precisión en entornos ruidosos	Muy buena, ajusta el filtro según ruido.	Excelente, ideal para entornos con ruido.	Buena si se elige bien el parámetro de suavizado.
Casos de aplicación	Restauración de imágenes, mejora de señales con ruido conocido.	Seguimiento de objetos (GPS, radares), robótica y control, fusión de sensores.	Análisis de series temporales, monitoreo financiero, control de calidad.
Ejemplos	Wiener en restauración de imágenes.	Kalman en navegación y robótica.	EWMA en control de procesos.

Tabla 2.3: Comparación entre distintos tipos de filtros digitales parte 2

De las Tablas anteriores, se puede observar que el filtro de Kalman ofrece mas ventajas sobre otros filtros para entornos dinámicos y con presencia de ruido por su capacidad de ajuste ante los cambios en la señal. Filtros como los Wiener y de mediana tienen un campo de aplicación mayormente hacia el procesamiento de imágenes, por otro lado los EWMA pese a su uso en sensores para filtrado de valores en tiempo real requiere de un coeficiente que puede ser asignado de manera empírica (similar al filtro Wiener) lo que puede provocar inestabilidad en el filtro. Los FIR y los IIR en procesamiento de voz, audio, en modulación y demodulación de señales de comunicación, entre otros.

Además de describir las características de los principales filtros digitales se presenta una simulación en la cual se demuestre la respuesta ante una señal contaminada por ruido.

El script que hace posible gráficar las respuesta de filtrado de la Figura 2.9 se encuentra en el anexo D.1. En donde, se tiene una señal original y en otra variable la señal original adicionándole ruido aleatorio. A la señal ruidosa se le aplican 4 filtros; primero un filtro FIR pasa bajos de orden 50, un IIR tipo Butterworth de orden 4, un filtro Wiener y un filtro de Kalman discreto utilizando las expresiones provistas por Simon (2006). Rápidamente se pude notar que la señal mas fiel a la original resulta ser la del filtro de Kalman siendo necesario solo aplicar las ecuaciones de predicción y corrección ademas de configurar el ruido R y Q.

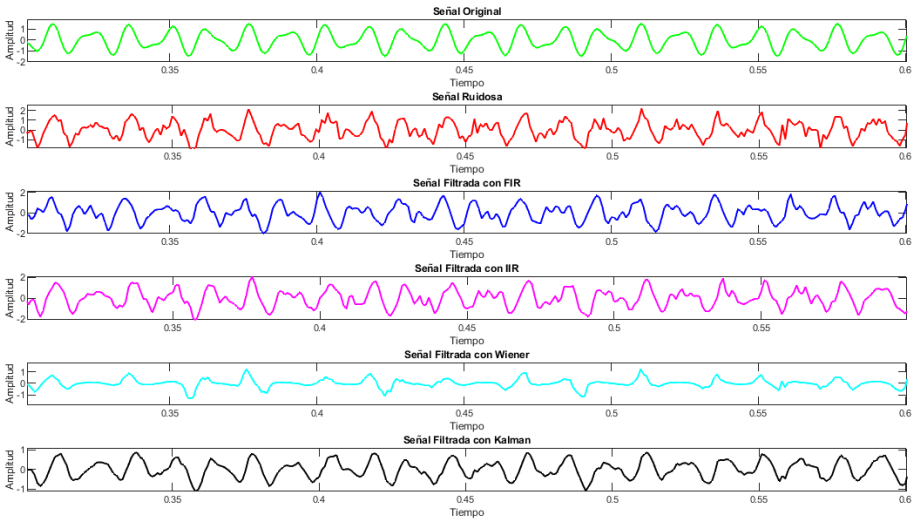


Figura 2.9: Simulación de filtros digitales en Matlab.

2.3.3. Filtro de Kalman en una dimensión.

De acuerdo a Simon [49] se establecen las expresiones matemáticas que estructuran al filtro de Kalman por medio de matrices, sin embargo, esto aplica cuando se tienen múltiples variables medidas.

En casos donde solamente se tiene una sola variable y no se tiene el modelo del proceso las ecuaciones son reducidas de manera que el calculo sea menos costoso a nivel computacional. Es aquí en donde se presenta al filtro de Kalman unidimensional, de acuerdo con Becker [6] puede implementarse con ayuda de ecuaciones que serán presentadas a continuación. Dichas ecuaciones tienen la capacidad de aplicarse en su forma discreta dando la posibilidad de programarlas en lenguaje de alto nivel.

Tal como se mencionó anteriormente, la predicción de estados es una de las etapas que comprende el filtro de Kalman y esto es dado que, a pesar de no conocer el valor del estado en un instante, si es posible realizar conjeturas acerca de el (en la teoría estadística se le conoce como el valor esperado). Las funciones de densidad de probabilidad plantean una forma de representar el valor esperado por medio del centro de la gaussiana de la distribución normal, mientras que la incertidumbre se refleja en el ancho de la gaussiana controlada por la desviación estándar o su cuadrado (la varianza). Por lo general, los errores de medición se distribuyen normalmente, por lo que para diseñar el filtro de Kalman se supone la distribución normal de los errores de medición.

En la Figura 2.10 se puede observar una distribución de funciones de densidad de probabilidad, el cual permite ilustrar la idea general del filtro de Kalman. Para ello se plantea un ejemplo sencillo, si partimos de una estimación acerca de la posición de un auto en el instante $\hat{x}_{k-1}$ y se tiene en cuenta el modelo matemático para obtener la posición del mismo, es posible realizar el cálculo y determinar una estimación preliminar en el paso siguiente de tiempo $\hat{x}_k$ (predicción). En ese mismo instante de tiempo el sensor que se utiliza para determinar la posición del auto arroja una medición y_k representada por otra función de distribución (naranja), sin embargo entre el cálculo de predicción y la medición se tienen valores diferentes, de aquí que el filtro de Kalman provee una estimación lo más optima posible unificando el estado de predicción con el obtenido por el sensor dando así un valor lo más cercano al estado real de la posición del auto (*Optimal state estimate* $\hat{x}_k$).

Al obtener una nueva función de distribución, el valor obtenido se acercará más al estado real de la posición del auto mientras más muestras se tengan a través del tiempo, de manera gráfica la nueva distribución normal deberá ser más alta y más angosta decreciendo la incertidumbre del estado.

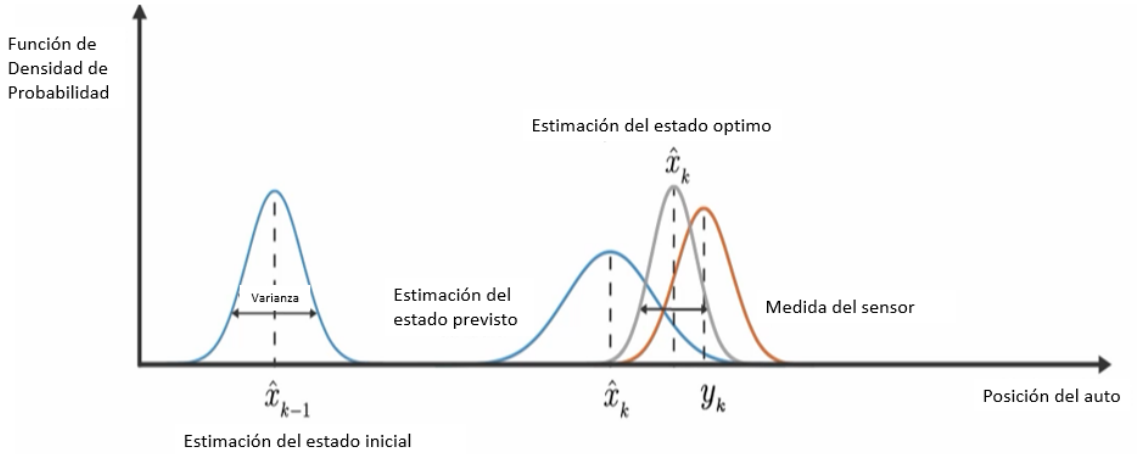


Figura 2.10: Distribución de funciones de densidad de probabilidad [5].

La primera ecuación del filtro de Kalman en una dimensión es la extrapolación del estado definido como:

$$\hat{x}_{n+1,n} = \hat{x}_{n,n} \tag{2.9}$$

Mientras que para la extrapolación de la varianza se emplea la siguiente expresión:

$$P_{n+1,n} = P_{n,n} + q_n \tag{2.10}$$

En donde:

- q_n = Varianza del ruido del proceso.
- $P_{n,n}$ = Actualización de la covarianza o bien la incertidumbre estimada del estado actual.

La ganancia de Kalman está definida por:

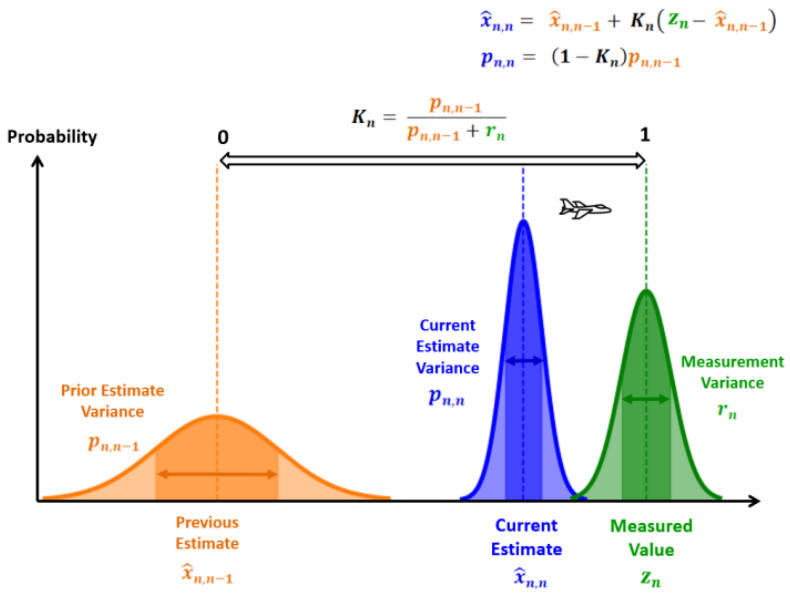
$$K_n = \frac{P_{n,n-1}}{P_{n,n-1} + r_n} \tag{2.11}$$

En donde:

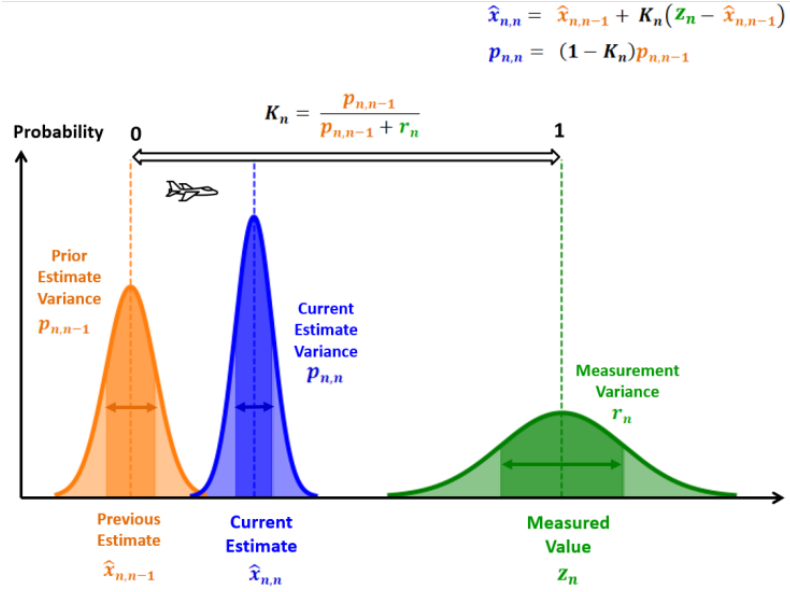
- q_r = Varianza de la medida.
- $P_{n,n-1}$ = Es la varianza de la estimación extrapolada o también conocida variación en la estimación.

Este parámetro debe estar dentro del rango mayor o igual a 0 y menor o igual a 1 de manera que cumpla con dos características de las funciones de densidad de probabilidad (deber ser positiva e igual a 1). Esta ganancia también es comúnmente conocida como factor de ajuste cuyo objetivo es minimizar la varianza del error se minimice. Iteración a iteración la ganancia de Kalman define el peso de la medición y

de la estimación anterior cuando se forma una nueva estimación indicando que tanto la medición modifica la estimación tal y como se puede observar en la Figura 2.11a. En 2.11a, si la incertidumbre de la medición baja en relación y la incertidumbre de estimación es alta la ganancia es cercana a 1 por lo que la nueva estimación se acerca a la medición, mientras que en 2.11b si la incertidumbre de la medición es alta y la incertidumbre de la estimación es baja la ganancia se acerca a 0 provocando que la nueva estimación se aproxime a la estimación anterior.



(a) Alta ganancia



(b) Baja ganancia

Figura 2.11: El efecto de la ganancia de Kalman [6].

La ecuación de actualización de estado se representa:

$$\hat{x}_{n,n} = \hat{x}_{n,n-1} + \alpha_n(z_n - \hat{x}_{n,n-1}) \tag{2.12}$$

En donde:

- $\hat{x}_{n,n}$ = La estimación del estado actual.
- $\hat{x}_{n,n-1}$ = El valor previsto del estado actual.
- α_n = Factor o también conocida como ganancia de Kalman K_n
- z_n = Medida obtenida.

Ecuación de actualización de la covarianza:

$$P_{n,n} = (1 - K_n)P_{n,n-1} \tag{2.13}$$

Las ecuaciones provistas por este autor son muy parecidas a las presentadas por Simon [49], el filtro de Kalman en una dimensión puede aplicarse a casos en los que únicamente se tiene una variable relacionada al proceso, en este caso se utilizara para "refinar" la medición de temperatura obtenida por el sensor. Al observar la ecuación de actualización de estado (2.12) inmediatamente se omite a la matriz H a comparación de la ecuación (2.5), mismo caso para la ecuación de la ganancia de Kalman y la actualización de la covarianza permitiendo aplicarse al sistema determinando su funcionamiento mediante iteraciones llevadas a cabo en tiempo real.

2.3.4. Aplicaciones del filtro de Kalman.

Los filtros de Kalman se utilizan frecuentemente para filtrar y suavizar las señales medidas en aplicaciones electrónicas, acondicionar señales de transductores, en sistemas de control para satélites, aeronaves y misiles. El filtro también se utiliza en aplicaciones además del control tal como la economía, la medicina, la química y la sociología.

Los filtros Kalman también se utilizan en cierta medida en el procesamiento de imágenes digitales, para mejorar la calidad de las imágenes digitalizadas. La detección posterior de señales en sistemas de radar y sonar y en sistemas de telecomunicaciones a menudo requiere filtros para la ecualización [50]. A continuación se presenta su implementación practica en Matlab.

Mitigación de errores en la medición de sensores.

Uno de los ejemplos más comunes de este filtro es la reducción de errores de medición, para mostrar de mejor manera esta aplicación se presenta por medio de la simulación en Matlab (programa en el anexo C.1). Para este caso de aplicación se parte de la simulación de un sistema para estimar la salida real de una planta en función de las mediciones con ruido. En Matlab se puede recurrir a la función Kalman,para diseñar el filtro en estado estacionario, una de la particularidad de

esta función es su capacidad para determinar la ganancia optima para una planta concreta en función de la covarianza del ruido de proceso Q y la covarianza del ruido del sensor R que se asigne. La planta modela en el anexo C.1 esta definida por la matrices A, B, C y D. Posteriormente, se crea el modelo de la planta con ruido de proceso por medio de la función *ss* contenida en la variable *sys*. Para este ejemplo se proponen ruidos de medida y de proceso constantes mayores a cero (Q=2.3 y R=1, aunque es importante obtener estos parámetros de acuerdo a mediciones directas o estudio previo del sistema).

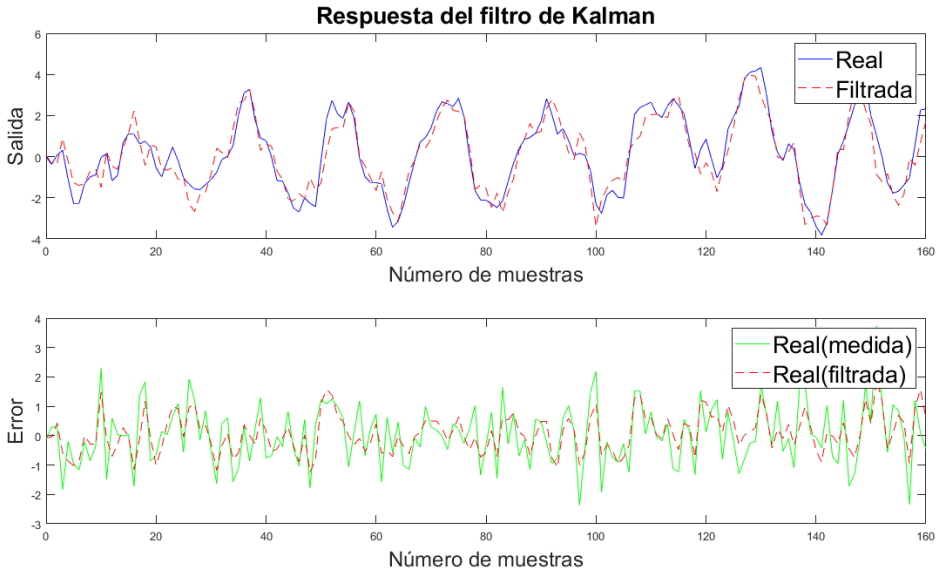


Figura 2.12: Mitigación de errores de medición

El efecto del filtro de Kalman se puede observar por medio de la Figura 2.12, en la cual se muestra de mejor manera la atenuación del error debido al error de medición.

El modelo de radar de posición de aeronaves.

Por otra parte, simulink ofrece modelos que permiten visualizar el propósito del filtro de Kalman (extendido), en esta ocasión enfocado a la estimación de la posición de una aeronave a partir del uso de mediciones de un radar. Cada estimación se guarda en el espacio de trabajo y se gráfican en secciones individuales tal como se presenta en la Figura 2.13.

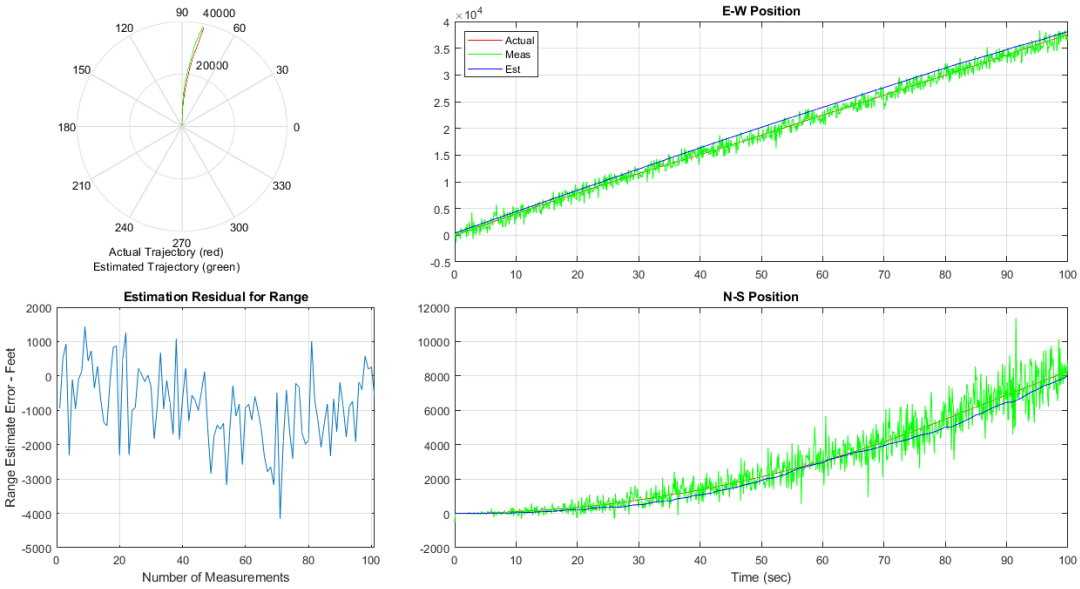


Figura 2.13: Estimación de la posición de una aeronave

2.4. Raspberry Pi

Una Raspberry pi puede ser considerada desde el punto de vista del usuario, una especie de mini ordenador que le permite realizar distintos proyectos gracias a las funciones que esta puede proveer, en la Figura 2.14 se presenta el modelo utilizado para el desarrollo del sistema.

Consta de una placa, un procesador, distintos tipos de pines y puertos de conexión. Esta placa, lanzada por la fundación Raspberry Pi (su primer modelo) en 2012 ha tenido variantes a lo largo del tiempo, sin embargo, puede ser acotada en 2 modelos; A y B. También se pueden encontrar otros modelos que si bien pueden diferenciarse por el tamaño o la adición de periféricos (como por ejemplo la Raspberry Pi 400) mantienen la misma arquitectura en sus modelos A y/o B.

Raspberry Pi OS o también conocido como Raspbian es el sistema operativo oficial de la placa. Esta *distro* Linux (distribución Linux) está basada en Debian, optimizado para funcionar en equipos ARM [7].

Figura 2.14: Raspberry Pi 3 B+ [7].

2.4.1. Hardware de Raspberry Pi

Los modelos que más prestaciones le aportan al usuario son los modelos A y B, y la principal diferencia entre ellos está en la conexión USB. Las versiones del modelo A consumen menos energía además de no poseer un puerto ethernet, caso contrario a las versiones del modelo B que si lo tienen. Una placa Raspberry estándar está conformada por los siguientes componentes:

- Memoria RAM
- Procesador (CPU)
- Procesador Gráfico (GPU)
- Conexión Ethernet
- Interfaz GPIO
- Toma XBEE (Comunicación inalámbrica)
- UART (Interfaz en Serie)
- Conexión para la fuente de alimentación.
- Conexión para hardware externo (Memoria microSD)

En donde, es necesaria la memoria microSD para el almacenamiento y arranque de la placa Pi. Desde el punto de vista de un usuario Windows la microSD vendría siendo el disco duro de la PC.

2.4.2. Modelos de Raspberry Pi.

En las Tablas 2.4 y 2.5 se presenta un resumen de las diversas versiones y modelos de la placa Raspberry Pi destacando la mejora que ha traído modelo tras modelo posicionando a esta tarjeta como una de las más versátiles y económicas en el mercado.

Es importante mencionar que hay dos formas de numerar los pines presentes en la tarjeta, estos son en modo GPIO BOARD y GPIO BCM. En donde, en modo GPIO BOARD se identifican de acuerdo al número impreso, es decir enumerando de izquierda a derecha tal y como se presenta en la Figura 2.15. Mientras que para el caso del modo GPIO BCM se identifican de acuerdo por el nombre GPIO, por ejemplo, en la Figura 2.15 se muestran pines en GPIO BCM como GPIO 2, GPIO 3, etc.

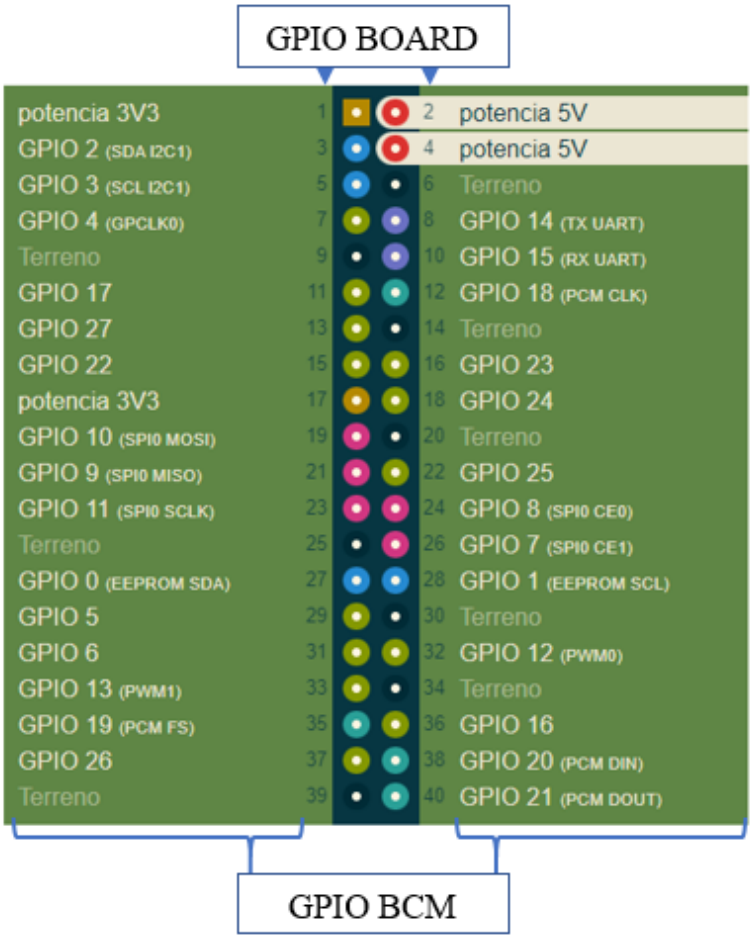


Figura 2.15: Numeración de pines de Raspberry Pi [8].

Model	RPI 3	RPI2	RPI B+	RPI A+
Price	$35	$35	$25	$20
Processor	BCM2837 quad core	BCM2836 quad core	BCM2835	BCM2835
Speed	1.2GHz	500MHz	700MHz	700MHz
Memory	1GB	1GB	512MB	256MB
Typical Power	2.5W (up to 6.5W)	2.5W (up to 4.1W)	1W (up to 1.5W)	1W (up to 1.5W)
USB Ports	4	4	4	1
Ethernet	10/100 Mbps Wi-Fi and Bluetooth	10/100 Mbps	10/100 Mbps	none
Storage	micro-SD	micro-SD	micro-SD	micro-SD
Video	HDMI	HDMI	HDMI	HDMI
GPU	Dual Core VideoCore TV Multimedia Co-Processor at 250MHz (24 GFLOPS)	Dual Core VideoCore TV Multimedia Co-Processor at 250MHz (24 GFLOPS)	Dual Core VideoCore TV Multimedia Co-Processor at 250MHz (24 GFLOPS)	Dual Core VideoCore TV Multimedia Co-Processor at 250MHz (24 GFLOPS)
Camera	yes	yes	yes	yes
GPIO header	40 pins	40 pins	40 pins	40 pins
Usage	General-purpose computing and networking. High-performance interfacing. Video streaming	General-purpose computing. High-performance interfacing. Video streaming	General-purpose computing. Internet connected host. Video streaming.	Low cost general-purpose computing. Standalone electronics interfacing applications.

Tabla 2.4: Comparación de Modelos Raspberry Pi (Parte 1)

Model	RPI ZERO	RPI B	COMPUTE
Price	$5	$25	$40
Processor	BCM2835	BCM2835	BCM2835
Speed	1GHz	700MHz	700MHz
Memory	512MB	512MB	512MB
Typical Power	1W (up to 1.5W)	1W (up to 1.5W)	1W (up to 1.5W)
USB Ports	1 OTG	2	Via header
Ethernet	none	10/100 Mbps	none
Storage	micro-SD	SO	4GB eMMC
Video	mini-HDMI	HDMI	HDMI via edge
GPU	Dual Core VideoCore TV Multimedia Co-Processor at 250MHz (24 GFLOPS)	Dual Core VideoCore TV Multimedia Co-Processor at 250MHz (24 GFLOPS)	Dual Core VideoCore TV Multimedia Co-Processor at 250MHz (24 GFLOPS)
Camera	no	yes	CSI x 2 via edge
GPIO header	40 pins	26 pins	48 pins via edge
Usage	Low cost small profile standalone electronics interfacing projects.	General- purpose legacy applications. Internet connected host.	Suitable for plugin into user-created PCB´s using a DDR2 SODIMM connector. Open-source breakout board available.

Tabla 2.5: Comparación de Modelos Raspberry Pi (Parte 2)

Parte fundamental de Raspberry Pi es la tarjeta SD (*Secure Digital*), que no es más que una tarjeta de memoria de dimensiones pequeñas diseñada para dispositivos electrónicos como los smartphones, ordenadores portátiles, consolas, cámaras de video, entre otros. Estas tarjetas se dividen en dos ramas:

- SD de almacenamiento: Orientadas al resguardo de imágenes, video, audio,

quemado de SO, etc.

- SD para transmisión: Funcionan exclusivamente como transmisores de entrada y salida de comunicación entre dos dispositivos.

Se necesita de una micro tarjeta SD de al menos 8GB de almacenamiento, en ella se debe quemar la imagen del sistema operativo a utilizar (Raspbian o alguna otra distro). Para este proyecto se utiliza una micro tarjeta SDHC de la marca Hyundai con las siguientes características:

- Almacenamiento de 16 GB.
- Dimensiones 15mm x 11mm x 1mm.
- Clase 10 (velocidad de escritura mínima de 10 MB/s).
- Velocidad de 12 MB/s para escritura.
- Velocidad de 25 MB/s para lectura.
- 8 pines.
- Voltaje de 2.7V a 3.6V.

Figura 2.16: Tarjeta SD Fuente: [9].

2.4.3. Señales PWM en Raspberry Pi

Una señal PWM (por sus siglas modulación de ancho de pulso) es considerado una señal de control de potencia suministrada a dispositivos electrónicos.

Funciona de la siguiente manera; se envían una serie de señales de encendido y apagado a una gran velocidad en lugar de enviar una cantidad de energía constante. El ancho de los pulsos determinara la cantidad de energía que se desea suministrar al dispositivo. Esta técnica se ve comúnmente en la atenuación de luminarias e inclusive en el control de motores.

Raspberry Pi en su modelo 3 B+ posee 4 pines que suministran señales PWM (unos más precisos que otros) los cuales se identifican como PWM0 y PWM1. La siguiente tabla muestra las salidas de la Raspberry Pi en su modelo 3 B+ relacionadas con señales PWM.

Pin GPIO	Número de pin	Función PWM	Descripción
GPIO 12	32	PWM0	Señal de reloj de propósito general
GPIO 13	33	PWM1	Señal de reloj de propósito general
GPIO 18	12	PWM0	Proporciona una señal de reloj a un dispositivo externo como un chip DAC.
GPIO 19	35	PWM1	Proporciona una señal de sincronización a un dispositivo externo con un DAC interno.

Tabla 2.6: Pines Raspberry Pi 3B+ que proveen pulsos de reloj.

2.5. Grove

Grove es un sistema para el desarrollo y creación de prototipos mediante el uso de conectores modulares y estandarizados. Uno de los principales objetivos de este sistema es la posibilidad de conectar y ensamblar de la manera más adecuada los distintos componentes electrónicos.

2.5.1. GrovePi

Grovepi es una librería integrada en el sistema operativo "*Raspbian for Robots*", el cual permite utilizar una gran variedad de módulos y sensores en conjunto con el armazón GrovePi+ compatible con Raspberry Pi 3B+. La librería utiliza el lenguaje de programación Python por defecto, aunque es posible manejar otros lenguajes como por ejemplo C, C++, NodeJS, Ruby, Scratch, Java. Es posible el uso de esta librería gracias al bus RPI-1SW resultado de la ejecución del bus I2C, implementado para evitar errores con el hardware I2C de Raspberry Pi. Las ventajas de esta forma de funcionamiento es la agilidad y el poco uso de recursos de la CPU. En cuanto al manejo de los puertos GPIO, se encuentran funciones generales como las que se presentan a continuación:

Función	Descripción	Valor de retorno
grovepi.digitalRead(pin)	Lee si la entrada de un puerto (D2-D8) está en un estado alto o bajo	0 o 1
grovepi.digitalWrite(pin, value)	Determina el valor de salida de un puerto digital (D2-D8), ya sea 0 o 1. El voltaje de salida es de 0V en el caso de establecer en 0 y 5V en caso de colocar en 1.	1 todo el tiempo.
grovepi.analogRead(pin)	Detecta una cierta cantidad de voltaje de acuerdo al puerto analógico seleccionado (A0-A2)	Un número de tipo entero de 10 bits que se asigna dependiendo del valor del voltaje de entrada del puerto.
grovepi.analogWrite(pin, value)	tilizado para configurar un nivel de voltaje siempre y cuando sea un puerto capaz de proveer señales PWM (D3, D5, D6, D9). Para el parámetro value se debe digital un numero de tipo entero que representara un numero de 8 bits con un valor de 0V a 5V.	1 todo el tiempo
grovepi.pinMode(pin, mode)	Función que sirve para configurar un puerto en modo de entrada o de salida. Lo puertos seleccionables van desde D2-D8. El modo entrado servirá para leer un valor y el de salida para escribir un valor o valores nuevos.	1 todo el tiempo.

Tabla 2.7: Ejemplo de tabla con 3 columnas y 4 filas

2.5.2. *Hat* GrovePi+

GrovePi+ es una plataforma de código abierto, un complemento o bien, un armazón/*hat* diseñado para la placa Raspberry Pi; utilizado para conectar una serie de sensores grove (relés, sensores de temperatura, pantallas OLED, detectores ultrasónicos, joystick, acelerómetros, sensores de humedad, GPS, entre otros) [51] en donde la comunicación entre los dos se produce a través una interfaz I2C. También es conocido como un marco de desarrollo que simplifica la interacción con sensores analógicos y digitales, haciéndolos "plug and play". Característica que permite la creación rápida de prototipos y la integración de varios módulos además de sensores lo cual es esencial para el monitoreo de los prototipos [52]. Todos los módulos GrovePi+ se interconectan por medio de conectores grove universales que constan de cables de 4 pines.

Los módulos grove funcionales con señales analógicas y digitales se conectan directamente al microcontrolador ATMEGA328 en GrovePi+. El microcontrolador actúa como un intérprete entre la Raspberry Pi y los sensores grove que envían, reciben y ejecutan comandos enviados por Raspberry Pi.

En los distintos proyectos que se han llevado a cabo con este *hat*, se ha observado la facilidad con la que se conectan y manejan los distintos módulos. Profundizando en la parte del código, Python resulta ser el lenguaje con el que mejor se puede interactuar evitando dificultades con el manejo de sensores y actuadores.

Es importante resaltar que es gracias a la distribución de *Raspbian for Robots* que es posible utilizar las funciones de GrovePi+, cualquier otra *distro* obstaculiza siquiera descargar el repositorio que contiene las dependencias y librerías necesarias para su conexión con Raspberry Pi.

El uso de GrovePi+ en los prototipos tiene implicaciones para el rendimiento, particularmente en términos de tiempo de procesamiento y consumo de energía. La integración de los sensores grove a través del GrovePi + permite una recolección y procesamiento eficiente de datos, lo cual es vital para las tareas de monitoreo que realizan las unidades.

2.5.3. Distribución de Puertos de GrovePi+

En la Figura 2.17 se presenta la manera en que están distribuidos los puertos y el tipo al que corresponde cada uno de ellos. Este complemento provee 7 puertos digitales, 3 analógicos, 3 I2C, 1 serial Grove, 1 serial Raspberry Pi además de los pines de alimentación y tierra (GND).

Entre los puertos ya mencionados también se pueden encontrar 2 filas de pines (26 en total), estos pines son u puente entre ambas placas, de manera que se aprovecha al máximo tanto los pines de la Raspberry como de la GrovePi+.

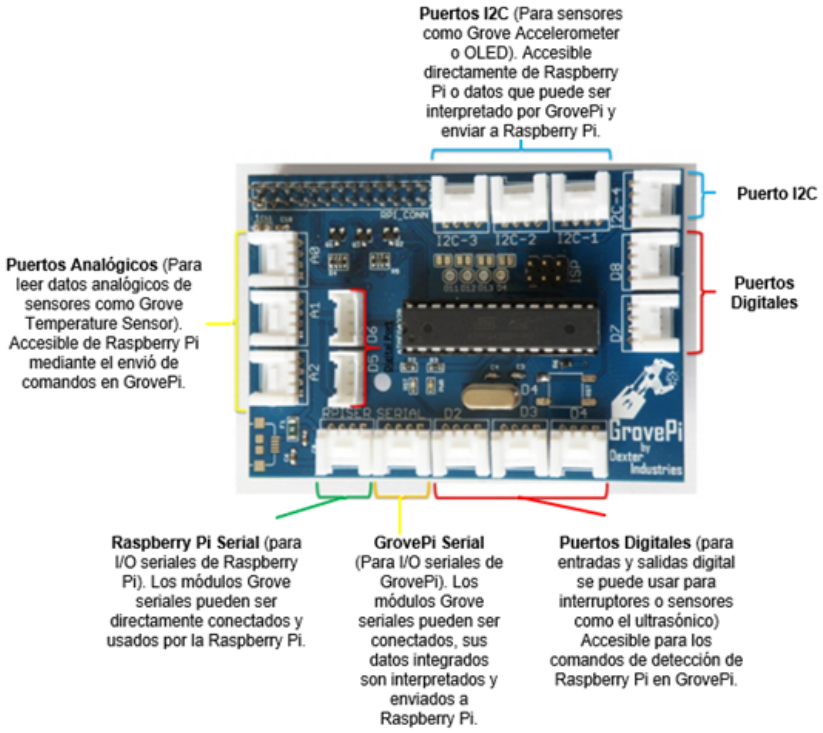


Figura 2.17: Puertos de GrovePi+.

2.5.4. Sensores

Instrumentos de entrada que miden alguna magnitud física para posteriormente brindar una señal de salida que pueda ser leída y procesada por algún operador. Son indispensables y relevantes en ámbitos de la seguridad, el monitoreo, la vigilancia, etc.

Los sensores desarrollan un papel fundamental en el campo de la automatización y en la industria, dado que llegan a ser componentes relevantes al brindar información del ambiente a sistemas para lograr un buen funcionamiento [53].

Se pueden clasificar según el tipo de variable que pueda leer, por el tipo de salida que proveen (analógica o digital), si es que llegan a influir en algún proceso (activos o pasivos), si son eléctricos, mecánicos, electromagnéticos, etc. Sin embargo, la manera más común de hacerlo es por el tipo de variable que pueda leer:

- Temperatura.

Miden la temperatura del ambiente en puntos específicos de alguna zona en específico o de un proceso. La manera de hacerlo es por medio de termocuplas, sensores fotoeléctricos, circuitos integrados que utilicen termistores, etc. El principal objetivo de los sensores de temperatura es medir que tan caliente o frio se encuentra una zona (entorno u objetos) para que se pueda realizar un ajuste (por poner un ejemplo relacionado con la automatización) o ejecutar alertas.

- Movimiento

Sirven para medir si un objeto se ha movido de posición, en este tipo de sensores es común el uso de IMUs (dispositivos que poseen en su interior acelerómetros y giroscopios), encoders (capaces de convertir el movimiento en una señal eléctrica), GPS.

- Distancia

También conocidos como transductores de distancia o transductores de posición lineal, sirve para medir la distancia de un objeto situado en una posición final. Los rangos de medida en estos sensores van desde los milímetros hasta los cientos de metros dependiendo de la tecnología utilizada (ópticos, inductivos, capacitivos, etc.).

- Luminosidad

Sensores que permiten medir la cantidad de luz en el ambiente, por lo general el componente principal que hace posible su funcionamiento radica en el uso de foto resistencias (LDR).

- Humedad

Miden la humedad en puntos específicos del ambiente, utilizados en el control de riego para cultivos son sensores de tipo capacitivos y resistivos que varían su valor dependiendo del grado de humedad que se presente en la zona de censado.

La forma de interconectar los sensores a utilizar en el sistema es por medio de cables universales de 4 pines, se tratan de conectores tipo macho, compatibles con todos los módulos Grove disponibles. Utilizados para conectar una placa base con un módulo (sensor, actuador) grove por lo que se puede considerar intermediarios de conexiones punto a punto. Estos cables se pueden encontrar en varias longitudes dependiendo de las necesidades del usuario, desde los 5cm hasta los 50cm.

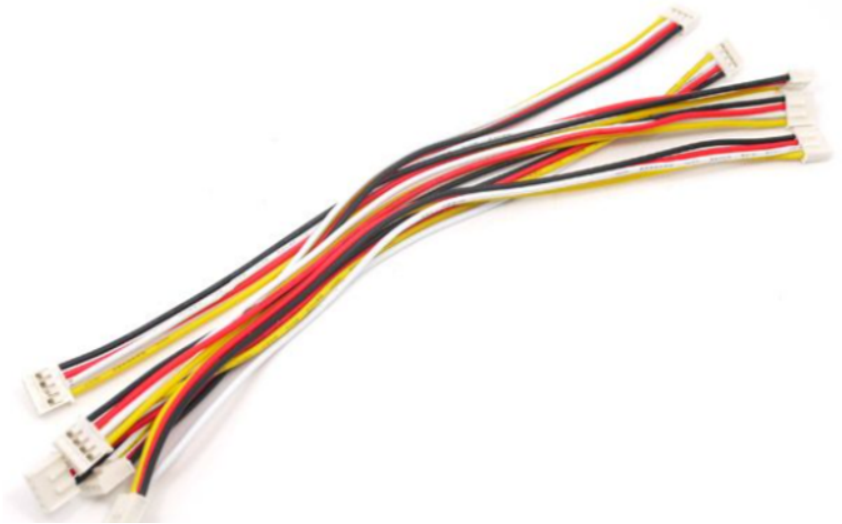


Figura 2.18: Cables universales de 4 pines Grove [10].

Los cuatro cables se encuentran codificados por colores, así como dos conectores de tipo macho localizados en cada extremo tal y como se muestra en la Figura 2.18.

Codificación de colores	Tipo de señal
Negro	GND (0V)
rojo	VCC(5V)
verde	Señal (SDA en conectores I2C por ejemplo)
amarillo	Señal (SCL en conectores I2C por ejemplo)

Tabla 2.8: Codificación de colores de los cables Grove de 4 pines.

Interfaces de los módulos Grove.
Los módulos grove tienen 4 tipos de módulos los cuales son:

- Digitales.
- Analógicos.
- UART.
- I2C.

Módulos grove digitales. Contiene dos líneas de señal denominadas comúnmente D0 y D1, y en la mayoría de los módulos grove digitales se llega a utilizar 3 de 4 pines; VCC, GND y D0.

Pin	Función
D0	Entrada/salida primaria digital
D1	Entrada/salida secundaria digital
VCC	Alimentación (3.3V-5V)
GND	Tierra

Tabla 2.9: Descripción de pines de los puertos digitales de GrovePi+.

Modulo grove analógico. Contiene dos líneas de señal denominadas comúnmente A0 y A1, en donde en la mayoría de los módulos analógicos se utiliza VCC, GND y A0.

Pin	Función
A0	Entrada/salida primaria analógica
A1	Entrada/salida secundaria analógica
VCC	Alimentación (3.3V-5V)
GND	Tierra

Tabla 2.10: Descripción de pines de los puertos analógicos de GrovePi+.

Modulo grove UART. Considerada una mejora del módulo grove digital, utiliza ambos pines de señal para la entrada y transmisión de datos en serie. El pin 1 se identifica como la línea RX y el pin 2 como TX. En donde RX es usada por la placa base para recibir datos y por ende manejándose como una entrada, mientras que la línea TX es utilizada por la placa para transmitir datos al módulo grove (salida).

Pin	Función
RX	Receptor serie
TX	Transmisor serie
VCC	Alimentación (3.3V-5V)
GND	Tierra

Tabla 2.11: Descripción de pines de los puertos UART de GrovePi+.

Modulo grove I2C. Estos módulos son una variante de los módulos grove digitales, el pin 1 es conocido como SCL mientras que el pin 2 SDA. El bus I2C actúa como un microcontrolador similar al ESP8266 o Arduino.

Pin	Función
SCL	Reloj I2C
SDA	Información I2C
VCC	Alimentación (3.3V-5V)
GND	Tierra

Tabla 2.12: Descripción de pines de los puertos I2C de GrovePi+.

Los módulos grove por lo general se presentan en 5 tamaños distintos, un ejemplo de ellos esta presentado en la Figura 2.19, mientras que el resto de los 4 módulos tienen dimensiones de 20x40 mm, 20x60 mm, 40x40 mm y 40x60 mm.

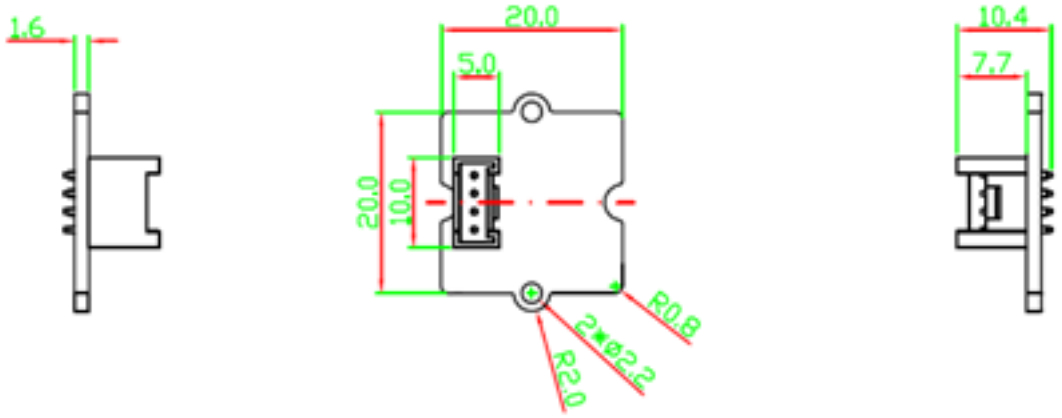


Figura 2.19: Diagramas generales de los módulos grove DIP 20mm x20mm [11].

2.5.5. Sensor de temperatura.

El sensor de temperatura *grove temperature sensor v1.2* realiza mediciones gracias a un termistor que cuenta con un alto nivel de estabilidad y precisión. El rango de operación de este sensor va de los -40° C a los 125° C con una precisión de ±1.5° C. Cuenta con un diseño que le permite ser "*Plug and Play*" permitiéndole ampliar la gama de aplicaciones con el uso de complementos para Raspberry Pi, tal como lo es GrovePi+ por poner un ejemplo.

Aplicaciones

- Domótica y control.
- Dispositivos IoT.
- Control Industrial.
- Monitoreo de clima.
- Sistemas de refrigeración y calefacción.

Especificaciones

- Dimensiones: 24mm x20mm x9.8mm
- Peso: 8g.
- Voltaje de operación: 3.3V-5V.
- Resistencia del termistor: 100KΩ
- Tolerancia: ±1 %
- Termistor; NCP18WF104F03RC (NTC)
- Rango de temperatura; -40° C – 125° C
- Precisión: ±1.5° C

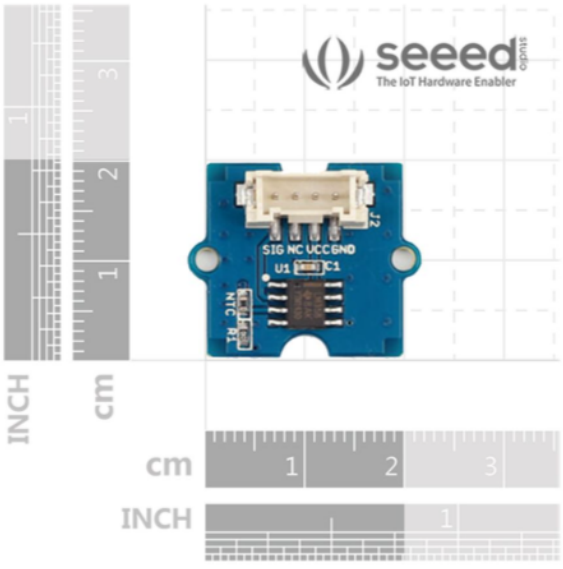


Figura 2.20: Grove temperature sensor V1.2 [12].

2.5.6. Sensor de humedad de suelo.

El sensor *Grove soil moisture sensor* realiza mediciones enfocadas a la humedad de suelo de las plantas. Este sensor está conformado por un par de sondas que permiten el paso de la corriente a través del suelo para poder obtener medidas de resistencia y así tener un valor en específico del nivel de humedad. Dado que también tiene la característica de ser "*Plug and Play*" es posible adaptarlo a prototipos enfocados a proyectos IoT, monitoreo, automatización, etc. Ya que la medida depende de un valor resistivo, mientras menor sea la resistencia mayor será el nivel de humedad, a un alto valor resistivo menor humedad de suelo.

Aplicaciones

- Jardinería botánica.
- Monitoreo de humedad.
- Medidas de consistencia.

Especificaciones

- Dimensiones: 60mm x20mm x6.35mm
- Peso: 10g.
- Voltaje de operación: 3.3V – 5V.
- Corriente de operación: 35mA.
- Valor de salida del sensor en suelo seco: 0-300
- Valor de salida del sensor en suelo seco: 300-700.
- Valor de salida del sensor en agua: 700-950.
- Tamaño de la PCB: 2.0cm x 6.0cm.

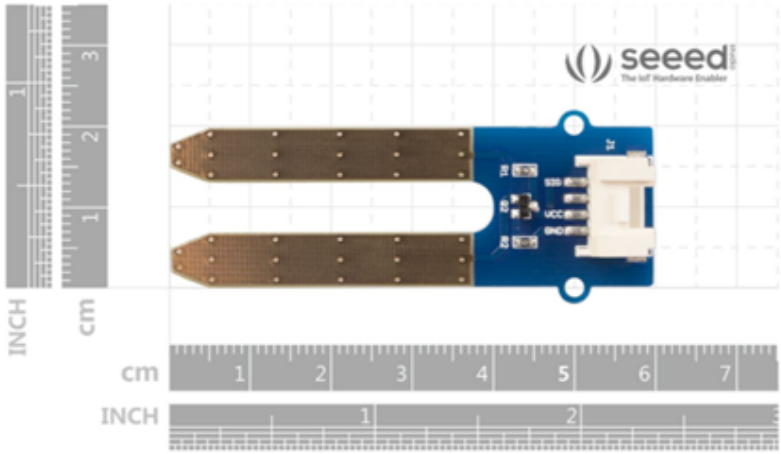


Figura 2.21: Grove moisture sensor [13].

2.5.7. Sensor de luz.

El sensor *Grove Light Sensor* es un módulo de tipo analógico capaz de generar señales eléctricas para convertirlas en distintos rangos. Uno de los componentes fundamentales en este sensor es el fotorresistor LS06-S, un foto diodo altamente sensible por el que es posible medir la cantidad de luz presente en el ambiente.

Otro de los componentes que forman parte de este sensor es el amplificador operacional dual LM358 cuya función es generar un nivel de voltaje dependiendo de la intensidad de la luz (basado en el nivel de la resistencia del LS06-S). El valor de salida del sensor es alto cuando se detecte un nivel alto de luz en el ambiente y bajo cuando más oscuro sea el nivel de luz.

Otra de las características, de las más destacadas es el tiempo de respuesta. Logrando obtener mediciones entre 20mS y 30mS. Este módulo también comparte la característica "*Plug and Play*".

Aplicaciones:

- Mediciones de luz.
- Detector de luz.
- Interruptor controlado por luz.
- Dispositivo domestico inteligente.

Especificaciones:

- Dimensiones: 20mm x20mm x10mm.
- Peso: 8g.
- Voltaje de operación: 3V – 5V.
- Corriente de operación: 0.5mA – 3mA.
- Tiempo de respuesta: 20mS – 30mS.
- Interfaz de entrada: Analógica.

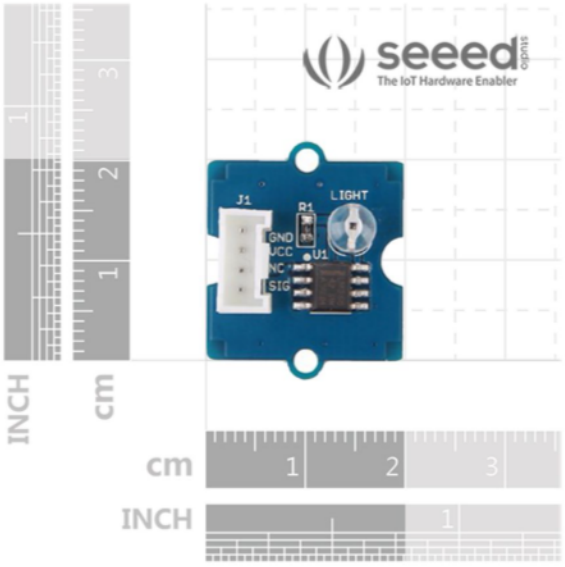


Figura 2.22: Grove Light Sensor [14].

2.6. Python

Generalmente el uso de Raspberry Pi se ha visto en aplicaciones basadas en Python, un lenguaje de programación de código abierto con un historial corto en el uso de GrovePi+ para la creación de proyectos.

2.6.1. Breve contexto histórico

Python es un lenguaje de programación ampliamente utilizado en aplicaciones web, el desarrollo de software, la ciencia de datos y el *machine learning* (ML). Algunas de las razones de utilizarlo es su eficiencia y facilidad de aprender, además de poder ejecutarlo en varias plataformas diferentes. Creado por Guido Van Rossum en 1989 publicando su primera versión en 1991 (Python 0.9.0), incluía características destacables como algunos tipos de datos y funciones para la gestión de errores.

La segunda versión (Python 1.0), lanzada en 1994 contenía nuevas funciones relacionadas con el procesamiento de listas de datos, asignación, filtrado y reducción. Posteriormente surgió Python 2.0 en el año 2000 brindándole al programador compatibilidad con los caracteres *Unicode* y una forma más corta de recorrer una lista. La tercera versión (Python 3.0) seria lanzada en diciembre de 2008, proveyendo entre las mejoras más sobresalientes la función de impresión, soporte para la división de números y gestión de errores. Es un lenguaje que aún no ha sido explotado en su totalidad además de contar con una popularidad ascendente en los últimos años, y no es para menos, ya que se ha utilizado en distintos proyectos involucrados con IoT. Con esto, las tarjetas de desarrollo basadas en este lenguaje han planteado alternativas de acuerdo a las necesidades de los usuarios.

2.6.2. Características principales del lenguaje.

Python es un lenguaje interpretado, esto quiere decir que no necesita de la compilación de los programas en cada modificación que se le hace al código. Esta es una de las características que le da una mayor ventaja sobre otros lenguajes de alto nivel como C o C++. En consecuencia la velocidad de desarrollo de aplicaciones aumenta [54]. Es un lenguaje tipeado dinámicamente, de manera que no es necesario definir el tipo de variable al escribir el código ya que Python determina automáticamente a que tipo se refiere en medio del tiempo de ejecución. Se trata de un lenguaje de alto nivel, por lo tanto, el programador no se debe preocupar por sus funcionalidades subyacentes como lo son la arquitectura y la administración de la memoria. Orientado a objetos, Python considera todo como un objeto, pero también puede admitir otros tipos de programación.

2.6.3. Interfaces gráficas de usuario

Una interfaz gráfica de usuario o también conocida como GUI (*Graphical User Interface*) es el conjunto de elementos y/o formas que en conjunto logran presentar al usuario una forma de interactuar con un dispositivo de manera que logre utilizar las funciones por las que fue desarrollado. El propósito de las interfaces gráficas es ofrecerle al usuario un entorno visual amigable y con la menor intervención de comandos. Una GUI presenta objetos que pueden realizar cambios o transiciones y mostrar información al momento de interactuar con el usuario, esto es, transiciones de colores, cambios de tamaño, opacidad y visibilidad en las formas. Entre las formas y elementos más conocidos están las ventanas, los iconos y los botones. También es posible representar datos de manera gráfica, por ejemplo, adicionando diagramas, gráficas de barras, de tipo pastel, etc. Otro de los elementos útiles son los campos de texto los cuales son una forma de alimentar de información a la aplicación o sistema basada en GUI.

2.6.4. Tkinter

Tkinter es una librería de GUI estándar del lenguaje de programación Python orientada a objetos. Posee herramientas que permiten la creación y administración de ventanas, cuadros de dialogo, botones, etiquetas, etc. Conocido como un kit de herramientas de código abierto compatible con múltiples plataformas. Esta librería hace posible la creación de widgets y aplicaciones con distintos propósitos con interfaces gráficas, idóneo para el desarrollo de aplicaciones de escritorio [54].

2.6.5. Matplotlib

Matplotlib es una librería de Python que tiene como objetivo la creación de gráficos en 2D y 3D; histogramas, espectros de potencia, gráficos de barras, diagramas de dispersión, gráficos de error, funcional para el análisis de datos de forma interactiva. Diseñado para trabajar en conjunto con otras librerías como Numpy (manejo de números, cálculos y procedimientos matemáticos), es completamente compatible con los kits de herramientas GUI que puede proveer Python como por ejemplo

Tkinter. Es posible adicionar gráficas en aplicaciones diseñadas en Tkinter dada su compatibilidad logrando una mejor presentación estética gracias a la API orientada a objetos que tiene integrada. Es la librería mas utilizada para en el desarrollo de proyectos en donde se requieran visualizaciones avanzadas, altamente eficiente [54].

2.6.6. XlsxWriter

XlsxWriter es una librería de Python que tiene como objetivo crear, editar y eliminar archivos en formato de Excel, por ende, es posible crear hojas de cálculo, adicionar datos mediante módulos de manipulación de celdas, agregar gráficos, etc. En cuanto a la manipulación de celdas, tiene la posibilidad de agregar información en formato de texto, números, generación de fórmulas, colocación de hipervínculos, entre otras funciones. Las aplicaciones de esta librería abarcan una amplia gama, ya que se puede emplear en campos como la automatización en la generación de informes por mencionar un ejemplo.

2.7. Control de la etapa de potencia del PID.

Para llevar a cabo el diseño de los circuitos electrónicos, se detallan en primera instancia los conceptos clave que determinan el funcionamiento de los componentes utilizados para el control de la etapa de potencia del controlador PID de temperatura.

2.7.1. Optoacopladores.

El optoacoplador es un dispositivo de aislamiento eléctrico, como su nombre lo dice tiene el objetivo de aislar partes de un circuito. Además, es capaz de transferir señales eléctricas entre las partes aisladas del circuito [55]. Realizan un acoplamiento entre la entrada y la salida por medio de emisión de luz, empelados en autómatas programables, en el control de potencia en corriente alterna (CA), etc.

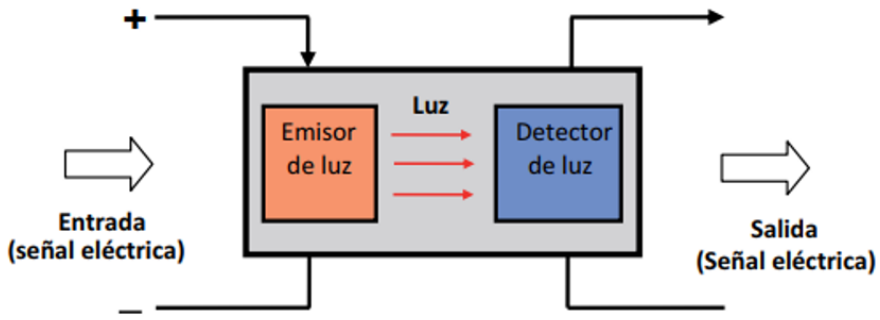


Figura 2.23: Estructura interna general de un optoacoplador [15].

Este elemento resulta ser una alternativa de solución a uno de los problemas en relación con la conexión de circuitos de control de baja tensión con otros dispositivos o elementos de alto voltaje. Si bien los transformadores también representan una

alternativa de solución, los optoacopladores por su tamaño pequeño y su eficiencia permiten tener un aislamiento más adecuado para las distintas aplicaciones que se le puede dar.

En la Figura 2.24 se ve representada la estructura del optoacoplador, el cual consta de un diodo LED y un fototransistor, en el momento que se polarice el LED, la radiación fotónica que emite acciona al fototransistor permitiendo el paso de la corriente en el colector-emisor.

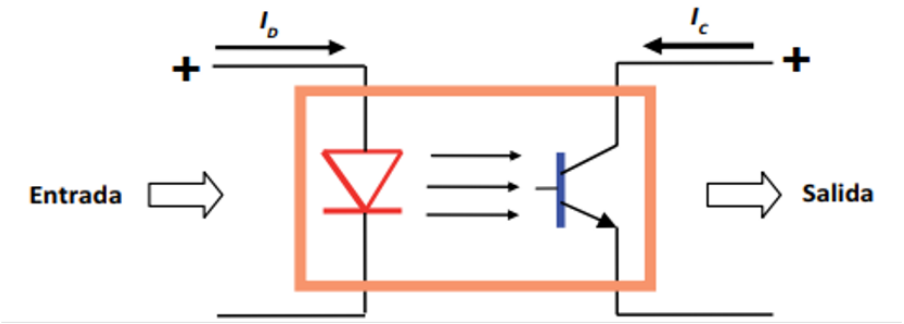


Figura 2.24: Circuito interno básico de un optoacoplador [15].

El optoacoplador MOC3021 forma parte de la serie MOC301XM y MOC302XM, considerado como un controlador de Triac con aislamiento óptico.

Características:

- Excelente estabilidad en la corriente del led emisor infrarrojo.
- Bloqueo de picos de tensión: 250V, MOC301XM, 400V, MOC302XM.
- Aprobaciones de seguridad y normativas: UL1577, 170 VAC_{RMS} por 1 minuto. DIN EN/IEC60747-5-5.

Aplicaciones

- Controles industriales.
- Controles de solenoides/válvulas.
- Semáforos.
- Interruptor de alimentación en CA.
- Máquinas expendedoras.
- Reguladores de intensidad de lámparas incandescentes.
- Relés de estado sólido.
- Control de motores.

- Balastros de lámparas.

Diagrama de pines

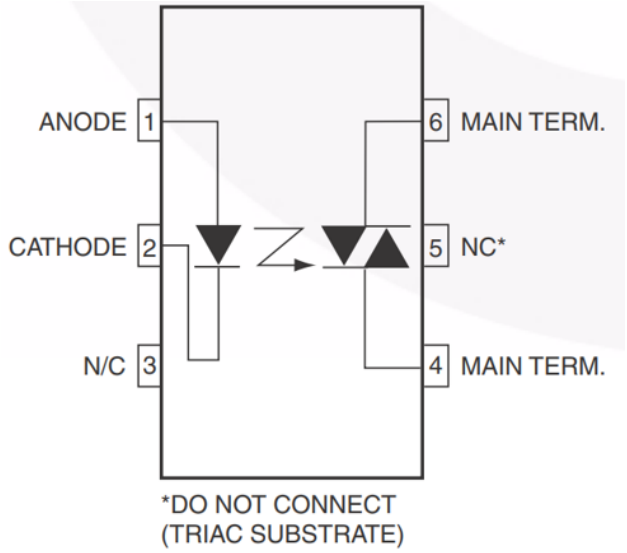


Figura 2.25: Esquemático [16].

A continuación, se presentan datos con relación a los valores nominales de entrada, salida y disipación del diodo y del foto Triac:

Valores de entrada del diodo	
Corriente directa	**50mA**
Voltaje inverso	**6V**
Potencia de disipación	**70mW**

Tabla 2.13: Valores nominales de entrada del foto triac.

Valores de entrada del diodo	
Voltaje del terminal de salida en estado desactivado	**400V**
Corriente directa (pico)	**1A**
Potencia de disipación	**300mW**

Tabla 2.14: Valores nominales de salida del foto Triac.

Detector de cruce de paso por cero.

El H11AA1 es un detector de cruce por cero (zero crossing) el cual emite un pulso cuando una señal de tensión pasa por 0V. Al trabajar con corriente alterna en la mayoría de las ocasiones resulta interesante realizar la detección del instante en el que la tensión atraviesa el punto cero. Suele ser de utilidad para medir la frecuencia de la red, para rectificar el desfase angular introducido por transformadores empleados al realizar la medición de la tensión o como es frecuente para determinar

el momento para conmutar una carga [56].Los detectores de cruce de paso por cero son ampliamente utilizados en aplicaciones que funcionan con corriente alterna abarcando tareas de medición e inclusive para realizar alguna acción. Es un componente conocido en aplicaciones de sonido y sistemas de comunicación.
La serie H11AA1 se identifica como aisladores acoplados de manera óptica, de tipo bidireccional conformados principalmente por dos diodos emisores de luz infrarroja conectados en paralelo y un foto transistor tipo NPN. Este C.I permite conectar una corriente alterna en el principio primario, mientras que el secundario se comportara como un interruptor cerrado excepto cuando la tensión en el primario inferior la tensión sea 0V, solo así es que se comportara como interruptor abierto.

A continuación se presentan algunas de sus características:

- Alto voltaje de aislamiento.
- Entrada de CA insensible a la polaridad,
- Parámetros eléctricos probados al 100
- Selecciones eléctricas personalizadas disponibles.

Aplicaciones:

- Terminales de computadora.
- Controladores de sistemas industriales.
- Dispositivos telefónicos, centrales telefónicas.
- Transmisión de señales entre sistemas de diferentes potenciales e impedancias.

Diagrama de pines:

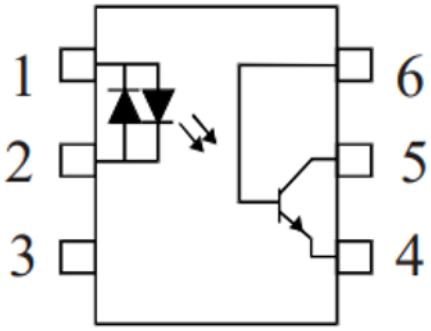


Figura 2.26: Esquemático [17].

A continuación, se presenta los valores nominales relacionados con este tipo de optoacoplador:

Valores máximos absolutos	
Almacenamiento de temperatura	**-55°C – 125°C**
Temperatura de operación	**-30°C – 100°C**
Temperatura de soldadura con plomo	**260° C**

Tabla 2.15: Rango de operación del optoacoplador H11AA2X.

Valores de entrada del diodo	
Corriente directa	**50mA**
Potencia de disipación	**70mW**

Tabla 2.16: Valores nominales de entrada del optoacoplador H11AA2X.

Valores de salida del transistor	
Voltaje colector-emisor	**35V**
Voltaje colector-base	**35V**
Voltaje emisor-colector	**6V**
Voltaje emisor-base	**6V**
Corriente de colector	**50mA**
Potencia de disipación	**150mW**

Tabla 2.17: Rango de operación del optoacoplador H11AA2X.

2.7.2. Control de potencia de una lampara en AC.

Una de las tareas relevantes para el control de temperatura del invernadero es el control de alimentación eléctrica de la bombilla térmica. Los elementos que realizan esta tarea es el Triac en conjunto con un dispositivo que arroje el disparo de control del "brillo" de la lampara, conectada a la línea de 120V AC tal y como se muestra en la Figura 2.27. En un sistema de control de onda completa, la cantidad de potencia que recibe la carga puede variar desde el 0 % (0°) hasta 100 % (180°).

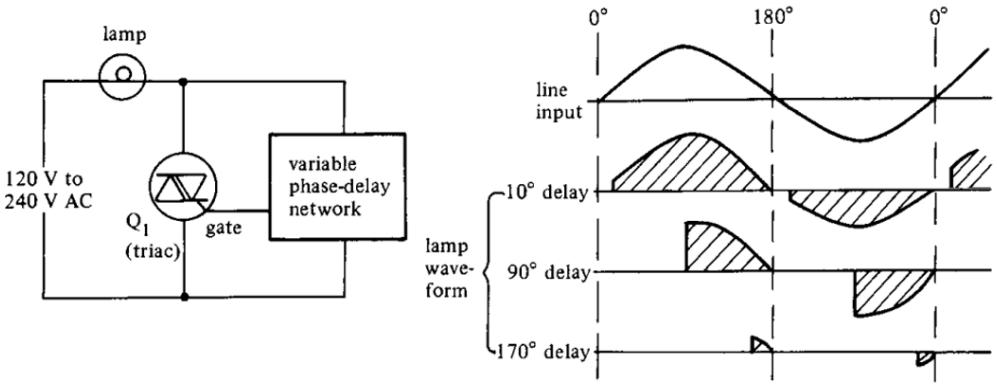


Figura 2.27: Circuito de control de luminosidad CA por disparo de fase y formas de onda [18].

Triac

El Triac es considerado un interruptor de alimentación bidireccional de estado sólido con auto enclavamiento con la capacidad de energizarse teniendo un impulso de disparo en su puerta (*Gate*) [18].

El Triac BT139 está diseñado para utilizarse en aplicaciones donde se requiera una transición alta de voltaje y de bloqueo bidireccional, así como un óptimo rendimiento cíclico. Este elemento que se puede observar como parte de sistemas como:

- El control de motores.
- Iluminación industrial.
- Calefacción.
- Conmutación estática.

Las características de este C.I son las siguientes:

- Capacidad de bloqueo de alto voltaje
- Activación directa desde controladores de bajo consumo y circuitos integrados lógicos.
- Puerta sensible (Gate)

Aplicaciones:

- Control de motores de propósito general.
- Conmutación de propósito general.

Tiene la capacidad de conectarse con microcontroladores, circuitos integrados lógicos y otros circuitos de bajo consumo.

PIN	Símbolo	Descripción
1	T1	Terminal 1
2	T2	Terminal 2
3	G	Compuerta de activación
mb	T2	Base de montaje, terminal 2

Tabla 2.18: Descripción de pines del Triac BT139.

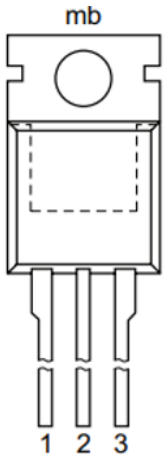


Figura 2.28: Encapsulado Triac BT139 [19].

A continuación, se presentan los valores nominales del Triac BT139.

Valores maximos absolutos	
Voltaje pico repetitivo fuera de estado	**600V**
Corriente RMS en estado encendido	**16A**
Corriente no repetitiva en estado encendido	**155A**

Tabla 2.19: Valores nominales del Triac BT139.

2.7.3. Calefactores térmicos

El componente que realizara la tarea de emitir calor dentro del invernadero es un bombillo cerámico o también conocido como calentador cerámico, el cual es un dispositivo que transfiere calor y se pueden clasificar en dos tipos:

- Calentadores infrarrojos de cerámica convectiva. El calor se transfiere pasando electricidad a través de estructura a base de aluminio.
- Calentadores infrarrojos de cerámica. Transfiere calor a través de cables de nicromo centrados en la cara del calentador para comenzar a irradiar. radiante.

Estos tipos de calentadores tienen como principio de calentamiento por medio de la resistencia eléctrica (energía eléctrica transformada en energía térmica), en donde, el calor generado es debido a la alta resistividad del calefactor cuando la energía eléctrica pasa a través de él [57]. De manera que la cantidad de energía eléctrica aplicada al calefactor es la misma que sale, pero en forma de calor. La estructura general de estos dispositivos se conforma de varillas, bobinas o cable de cerámica expuesto de una aleación. Presentan ventajas como:

- Calentamiento rápido.
- Seguros y portátiles.
- Eficientes en cuanto a consumo eléctrico.
- Alta vida útil.

Los calefactores se encuentran clasificados por tipos, los más comunes son:

- Tipo panel.
- Tipo varilla.
- Tipo tornillo.
- Tipo variante.
- Especiales.

El actuador que forma parte del controlador PID de temperatura es un calefactor cerámico de tipo tornillo debido a que posee las siguientes características:

Calefactor cerámico	
Modelo	ND-01
Voltaje	110V AC
Watts	100W
Conexión de entrada	E27 Tipo tornillo

Tabla 2.20: Especificaciones del calefactor cerámico

Figura 2.29: Bombillo cerámico ND-01 [20].

2.7.4. Amplificadores operacionales.

Otro de los elementos necesarios para el circuito de control de potencia es el amplificador operacional, se trata de un C.I que hace posible el diseño y construcción de circuitos electrónicos como por ejemplo los filtros, generadores de onda, amplificadores de audio, etc ([15]). Generalmente son representados por medio del siguiente diagrama:

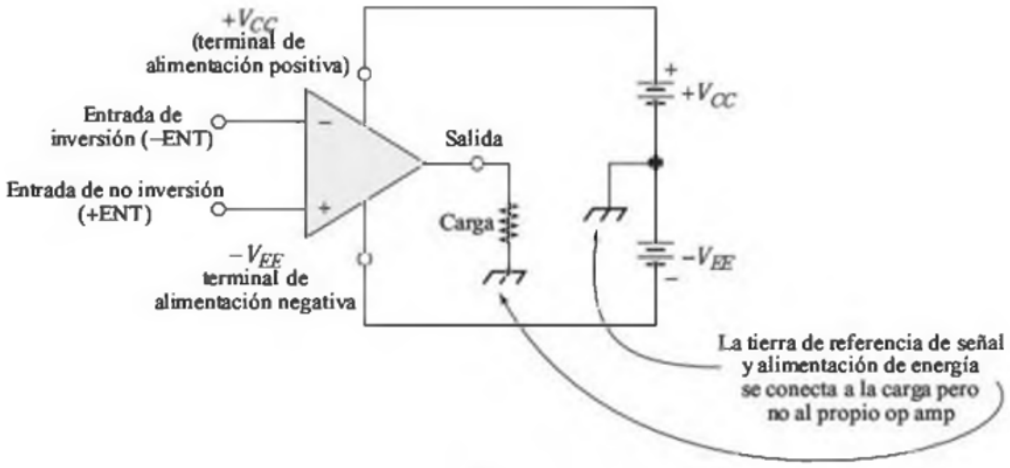


Figura 2.30: Terminales de un amplificador operacional [21].

También conocido como "op amp" consta de 5 terminales, y tienen las siguientes características:

- Ganancia.

Señal de entrada con el efecto de amplificación, esta denotada por la expresión:

$$A_v = \frac{v_o}{v_i} \tag{2.14}$$

Donde:

A_v= Ganancia.
v_o= Tensión de salida.
v_i= Tensión de entrada.

- Impedancia de salida. Resistividad interna en la terminal de salida del amplificador operacional.
- Impedancia de entrada. Resistividad conectada en serie a la entrada del amplificador operacional.
- Ancho de banda. Es el conjunto de frecuencias que se amplifica sin que la señal de salida disminuya más de 3dB (decibelios).
- Son amplificadores diferenciales. Dado que estos amplificadores cuentan con dos entradas de señal, se amplifica la diferencia de tensión de ambas señales.
- Alimentación simétrica. En la Figura 2.30 se puede observar la presencia de 2 terminales de alimentación del op amp, una de ellas recibe tensión positiva y la otra negativa permitiendo que la señal de salida varié de un valor negativo a positivo.
- Entrada no inversora. La polaridad de la señal en esta entrada se obtendrá en la salida con la misma polaridad, no se produce un efecto de inversión de fase.
- Entrada inversora. La polaridad de la señal aplicada en la entrada inversora se manifiesta en la salida con la polaridad contraria, se produce un efecto de inversión de fase.

Configuraciones de los amplificadores operacionales.

A continuación, se presenta cada una de las configuraciones de estos circuitos que resultan ser los más prácticos.

- Seguidor de tensión.

Circuito comúnmente utilizado como adaptador de impedancias. No amplifica la señal de la entrada y el voltaje de salida es muy cercano al de la entrada.

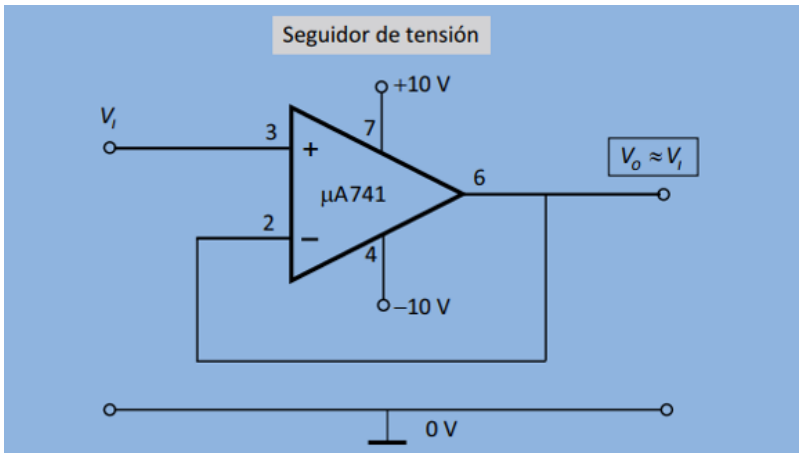


Figura 2.31: Amplificador operacional seguidor de tensión [15].

$$A_v = \frac{v_o}{v_i} \approx 1 \tag{2.15}$$

- Amplificador inversor.

Circuito cuya finalidad es la de amplificar la tensión de entrada, de lazo cerrado y con realimentación negativa. La expresión matemática que corresponde al cálculo de la ganancia está dada por la ecuación 2.16.

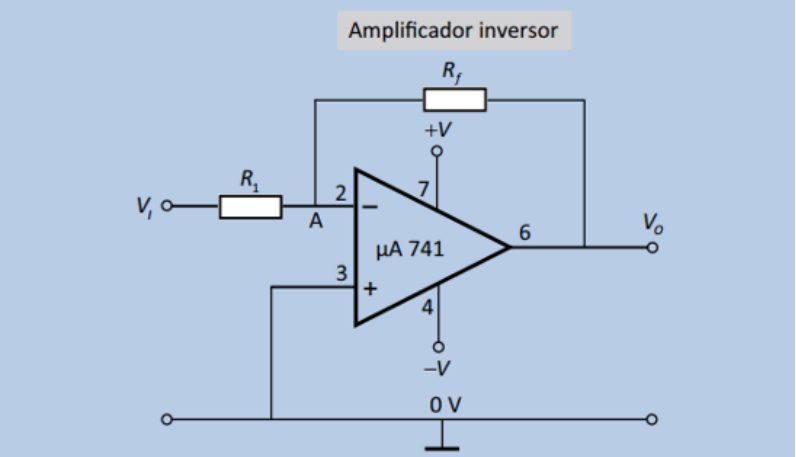


Figura 2.32: Amplificador operacional inversor Fuente: [15].

$$A_v = -\frac{R_f}{R_1} \tag{2.16}$$

- Amplificador no inversor.

Circuito que además de amplificar la señal de entrada, tiene la particularidad de no invertir la polaridad de la señal de salida, se logra conectando la señal de entrada en la terminal no inversora.

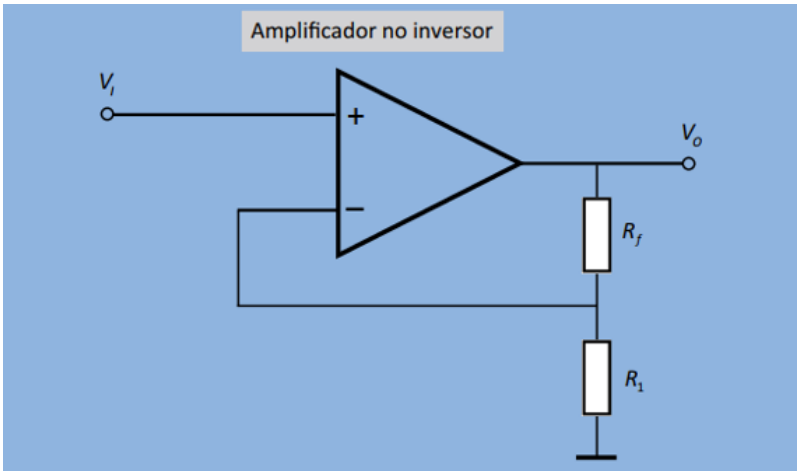


Figura 2.33: Amplificador operacional no inversor [15].

$$A_{vf} = \frac{R_f + R_1}{R_1} = 1 + \frac{R_f}{R_1} \tag{2.17}$$

- Circuito sumador.

Este circuito tiene la estructura de un amplificador inversor con la diferencia en la cantidad de entradas que posee. Con esta adecuación, a la salida del circuito se tiene el resultado de la suma de las señales de la entrada. Hay una inversión de fase.

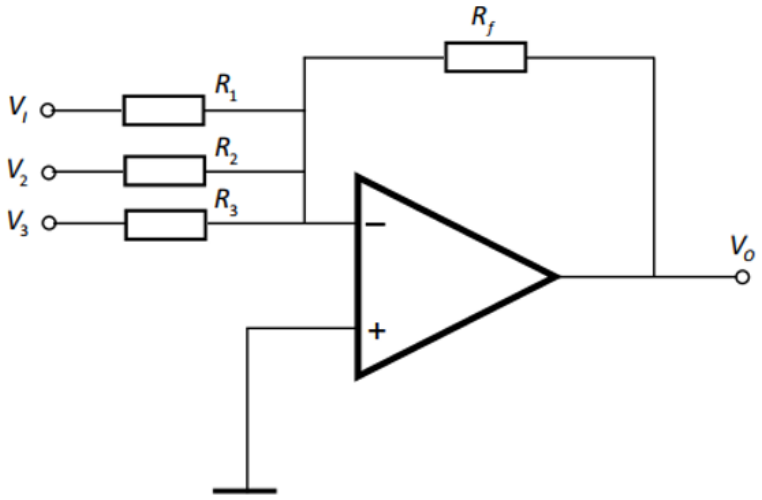


Figura 2.34: Circuito sumador [15].

$$V_o = -(\frac{R_f}{R_1}V_1 + \frac{R_f}{R_2}V_2 + \frac{R_f}{R_3}V_3) \tag{2.18}$$

- Circuito restador.

En este circuito la tensión de salida es proporcional a la tensión diferencial de las entradas, que a su vez es amplificada dependiendo de la resistencia en realimentación negativa.

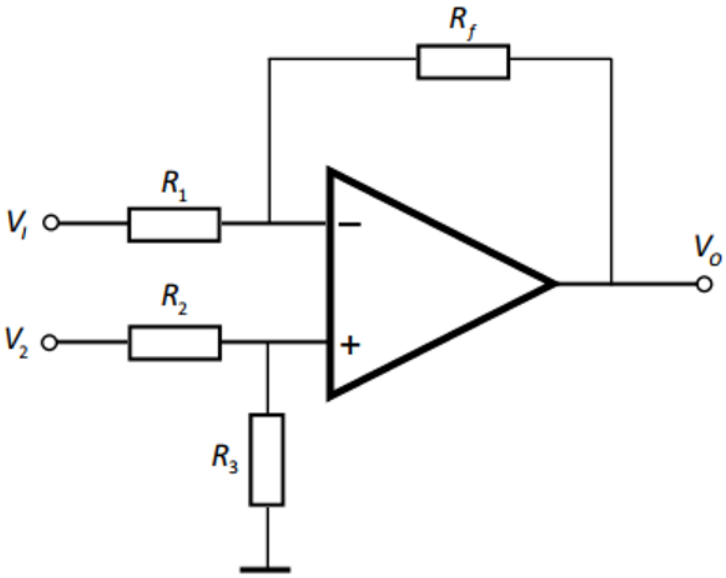


Figura 2.35: Circuito restador [15].

$$V_o = \frac{R_f}{R_1}(V_2 - V_1) \tag{2.19}$$

El modelo del circuito a utilizar es el TL084CN que forma parte de la serie TL08xH (TL081H, TL082H y TL084H). Esta serie ofrece una alta velocidad de respuesta (20V/μS) así como una entrada común para la alimentación. Además, es un componente con una alta resistencia a descargas electrostáticas.

En cuanto a su operación, funciona en un rango de -40° C a 125°C idóneas para aplicaciones más robustas.

Características:

- Alta velocidad de respuesta.
- Bajo ruido.
- Circuito de protección de salida.
- Baja distorsión armónica total.
- Amplio voltaje de suministro.

Aplicaciones:

- Energía solar.
- Control de motores.
- Trifásico UPS.

- Mescladores de audio profesional.
- Equipos de prueba de batería.

Pin	No.	Tipo Descripción
1IN-	2	Entrada inversora
1IN+	3	Entrada no inversora
1OUT	1	Salida
2IN-	6	Entrada inversora
2IN+	5	Entrada no inversora
2OUT	7	Salida
3IN-	9	Entrada inversora
3IN+	10	Entrada no inversora
3OUT	8	Salida
4IN-	13	Entrada inversora
4IN+	12	Entrada no inversora
4OUT	14	Salida
VCC-	11	Alimentación negativa
VCC+	4	Alimentación positiva

Tabla 2.21: Descripción de pines del amplificador operacional TL084CN.

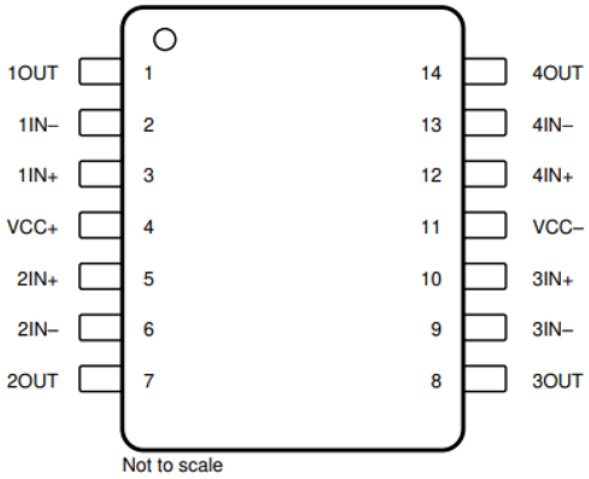


Figura 2.36: Diagrama de pines [22].

Fuente de alimentación S-60-12

Para implementar un amplificador operacional es imprescindible contar con fuentes de alimentación capaces de suministrar +12V y -12V, para ello se utiliza un par de fuentes de alimentación conmutadas con aislamiento y alta resistencia eléctrica. La fuente de alimentación modelo S-60-12 contiene una carcasa de metal perforada que hace más eficiente la disipación de calor. Cuenta con un amplio rango de voltaje de entrada CA que va desde 85V AC hasta 264V AC, provee 12V DC de salida con

una corriente máxima de 5A.

Esta fuente tiene una amplia gama de aplicaciones en el campo de la automatización industrial; en impresoras 3D, en la implementación de sistemas de CCTV, para amplificadores de audio, etc.

En la Figura 2.37 se puede observar su estructura, la cual está conformada por 3 borneras exclusivamente para la línea eléctrica (línea, neutro y tierra) y dos más para la salida en corriente continua (positivo y negativo) que suministra +12V.

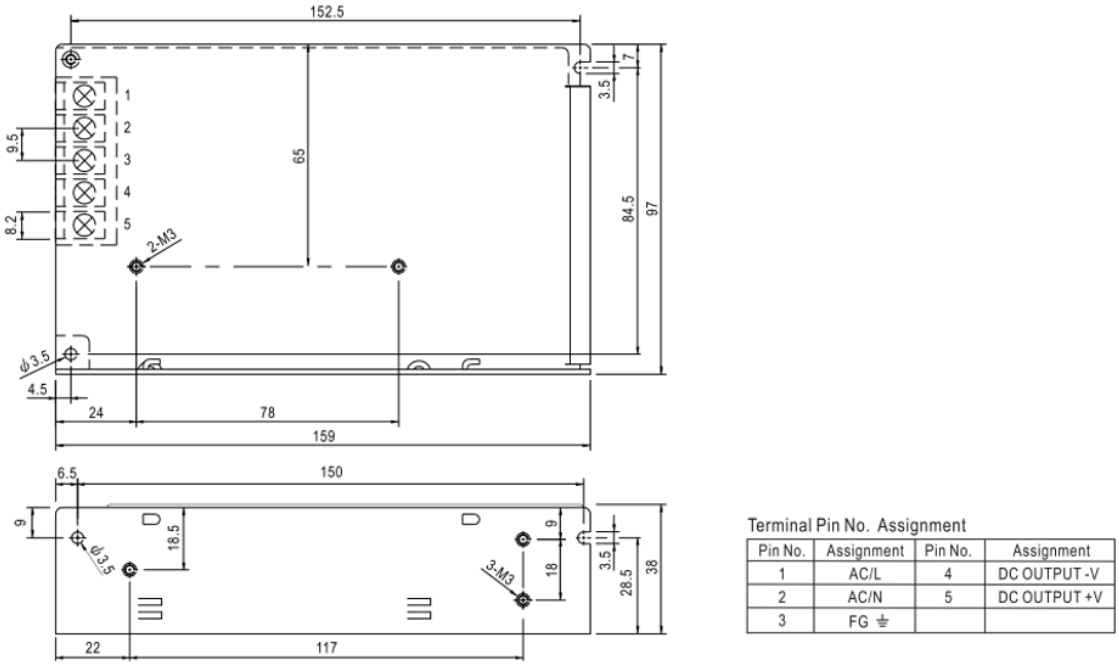


Figura 2.37: Fuente de voltaje modelo s-60-12 [23].

Especificaciones:

Entrada	
Rango de voltaje	85-264 VAC
Rango de frecuencia	47-63 Hz
Eficiencia	76 %
Corriente AC	2A/115 VAC, 1A/230 VAC

Tabla 2.22: Valores nominales de entrada de la fuente de alimentación S-60-12.

Salida	
Voltaje DC	12V
Corriente	5A
Rango de corriente	0A-5A
Potencia	60W
Rango de voltaje	10.8V DC–13.2V DC
Tolerancia de voltaje	± 1.0 %

Tabla 2.23: Valores nominales de entrada de la fuente de alimentación S-60-12.

2.8. Thonny IDE.

Thonny IDE es un entorno de desarrollo que tiene como objetivo el aprendizaje de la programación Python. Contiene características las cuales abarca desde la forma de visualizar la ejecución del código, así como el estado de las variables apoyando el proceso de análisis de la programación del usuario. Se trata de un software de código abierto desarrollado con Python 3, opera por medio de ventanas gráficas desarrolladas con la librería Tkinter y código *backend* para la ejecución de los scripts del usuario. Permite la adición de extensiones sin alterar negativamente la compilación de los programas [58]. Este entorno desarrollo es utilizado para generar y compilar el sistema de monitoreo dado que viene integrado en el sistema operativo *Raspbian for Robots*. En la Figura 2.38 se logra visualizar el espacio de trabajo.

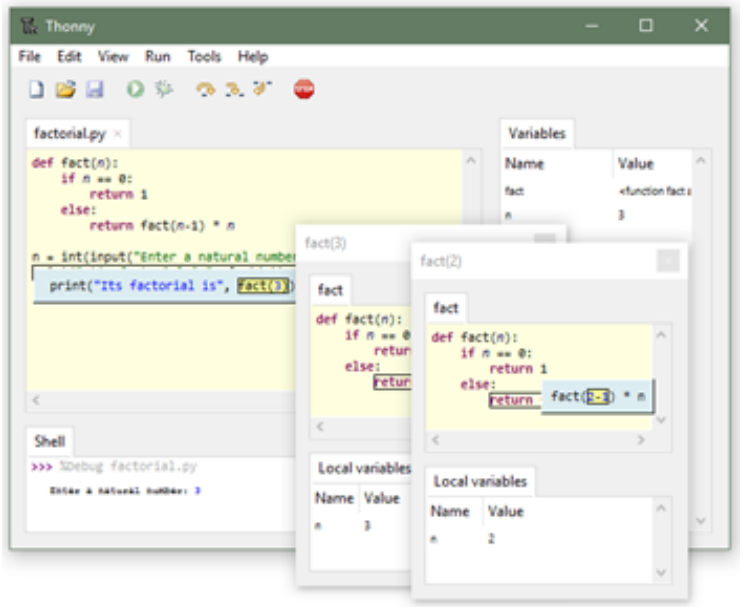


Figura 2.38: Entorno de programación Thonny IDE [24].

2.9. Sistema operativo *Raspbian for Robots.*

Raspbian for Robots es un sistema operativo basado en Raspbian creado por la compañía Dexter Industries, destinado a kits de robots basados en Raspberry Pi. Este sistema operativo contiene el software integrado para conectarse a GoPiGo, BrickPi, GrovePi o Arduberry con el fin de diseñar, desarrollar y programar prototipos con distintos fines. Los entornos de programación como Scratch y Python ya disponen de scripts que permiten entender el funcionamiento básico de los módulos y sensores grove, esto es, instrucciones en el lenguaje de Python útiles para manejar módulos Grove (sensores, botones, pantallas LCD, relevadores, etc.).

Figura 2.39: Tarjeta SD con el sistema operativo Raspbian for Robots [25].

Capítulo 3

Diseño y construcción.

Este capítulo aborda la construcción del prototipo del invernadero, elaboración de los circuitos realizados en Proteus y simulaciones en LTSpice para su implementación en las tablas fenólicas, así como el análisis matemático necesario para la obtención de las ganancias del controlador PID. También se incluye el diseño del sistema de monitoreo y los elementos requeridos en la programación.

3.1. Diseño del invernadero

Se utiliza una herramienta de diseño en 3D para el modelo de un invernadero casero con el fin de tener el espacio en donde se montará el sistema de monitoreo y por ende la implementación del controlador PID de temperatura. En el diseño mostrado a continuación se pueden observar dos niveles los cuales servirán para distribuir los componentes que se utilizaran a lo largo de este proyecto. El material utilizado es principalmente a base de PVC, dicho material es seleccionado por su alta resistencia a la humedad que se puede llegar a generar con el riego, las condiciones de temperatura y el mismo material con el que se forra la estructura del invernadero.

En la Tabla 3.1 se puede mostrar el desglose de componentes según la base a la que corresponde.

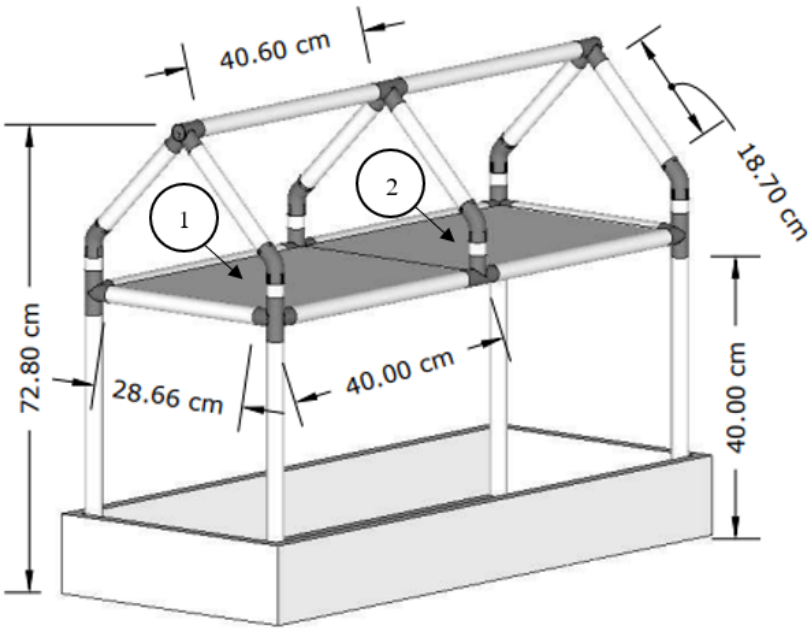


Figura 3.1: Render del invernadero.

En la Figura 3.1 y 3.2 se encuentran indicadores enumerados del 1 al 3 para cada base (o bien una tabla de apoyo), cada indicador sirve de apoyo para ubicar los distintos elementos del sistema.

Las dimensiones que se muestran en ambas ilustraciones son las adecuadas considerando que es el prototipo en el cual se aplicara el sistema de control (ademas del PID de temperatura y filtro de Kalman).

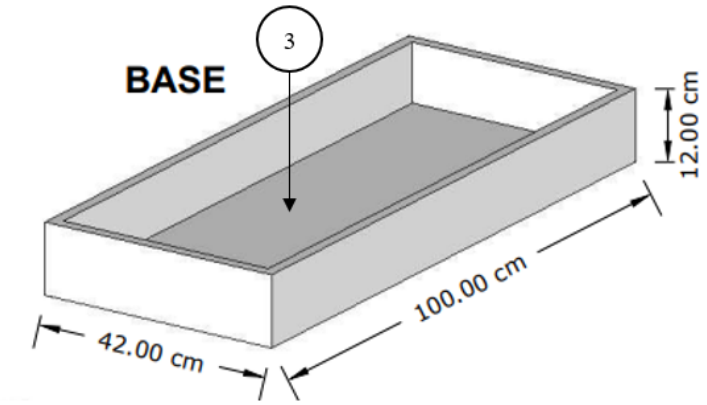


Figura 3.2: Base del invernadero.

De acuerdo con la Tabla 3.1 el nivel 2 contendrá la mayor parte de los componentes electrónicos, mientras que en el nivel 1 el espacio se destina a los cultivos ademas de los sensores.

Número	**Etiqueta**	**Descripción**
1	Base A N2	Primera base para distribución de componentes.
2	Base B N2	Segunda base para la distribución de componentes.
3	Base A N1	Base principal en la que se posicionaran elementos de adquisición de datos y el elemento actuador de temperatura.

Tabla 3.1: Simbología de bases del invernadero.

En base con la Figura 3.1 se presenta el siguiente concentrado relacionado con el material necesario para la construcción futura del invernadero. Recordando que la mayor parte de los materiales son a base de PVC (tubos y conectores) y madera.

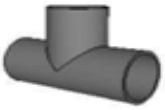
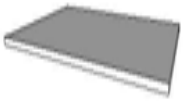
Tipo	**Longitud c/u**	**Medida ϕ**	**Cantidad de piezas**	**Vista previa**
Tubería PVC	40.00 cm	$\frac{1}{4}$”	8	
	28.66 cm	$\frac{1}{4}$”	2	
	2.00 cm	$\frac{1}{4}$”	6	
	18.70 cm	$\frac{1}{4}$”	6	
	40.60 cm	$\frac{1}{4}$”	2	
Codo PVC	N. A	45°	6	
Conector 4 vías PVC	N. A	45°	1	
Conector 3 vías PVC	N. A	$\frac{1}{4}$”	3	
Conector 4 vías PVC	N. A	$\frac{1}{4}$”	4	
Conector en T PVC	N. A	$\frac{1}{4}$”	2	
Base número 1	45cm x 32cm x 1cm	N. A	1	
Base número 2	45cm x 32cm x 1cm	N. A	1	

Tabla 3.2: Lista de componentes de PVC

De acuerdo con cada tabla de madera identificada por etiquetas, es en la Tabla 3.3 que se presenta la distribución de los componentes específicos que abarcan al sistema de monitoreo.

Base A N1	**Base A N2**	**Base B N2**
Descripción	Descripción	Descripción
Desglose de componentes asignados a la placa o base “A” correspondiente al nivel 1.	Desglose de componentes asignados a la placa o base “A” correspondiente al nivel 2.	Desglose de componentes asignados a la placa o base “B” correspondiente al nivel 2.
Componentes	Componentes	Componentes
Sensor de temperatura *grove Temperature Sensor V1.2*, sensor de luz *grove Light Sensor*, sensor de humedad *grove Humidity Sensor*, cables universales de 4 pines, planta o cultivo, bombilla cerámica emisora de calor, sistema de lámparas de luz de crecimiento para plantas.	Raspberry Pi 3 B+, GrovePi+, circuito detector de cruce por cero, circuito de Potencia, unidad de multi contactos para alimentación de dispositivos, fuentes de alimentación a 12V DC.	Display de 7 pulgadas.

Tabla 3.3: Distribución de componentes en el invernadero.

La distribución observada en la Tabla 3.3 está organizada considerando el espacio aproximado de cada uno de los componentes con el fin de no saturar las dimensiones de cada una de las bases.

Una vez descrito modelo del invernadero y la distribución de los componentes, el paso siguiente es definir el sistema de monitoreo. Para tal propósito se muestra con apoyo de la Figura 3.3 la estructura general considerando el hardware necesario y su interconexión con los distintos módulos utilizados.

3.2. Estructura general del sistema de monitoreo.

Los componentes principales van desde módulos, sensores, circuitos necesarios para el controlador PID, fuentes de alimentación y los pines/puertos requeridos para su completo funcionamiento. GrovePi+ es una pieza fundamental en la elaboración de este sistema ya que será útil al momento de comunicar el programa principal de las interfaces gráficas con el hardware en los puertos del hat, del diagrama de la Figura 3.3 también se observa el uso de los pines de la Raspberry Pi, más en específico el pin 18 cuyo propósito es arrojar una señal de reloj para configurar la frecuencia de trabajo en tiempos muy precisos.

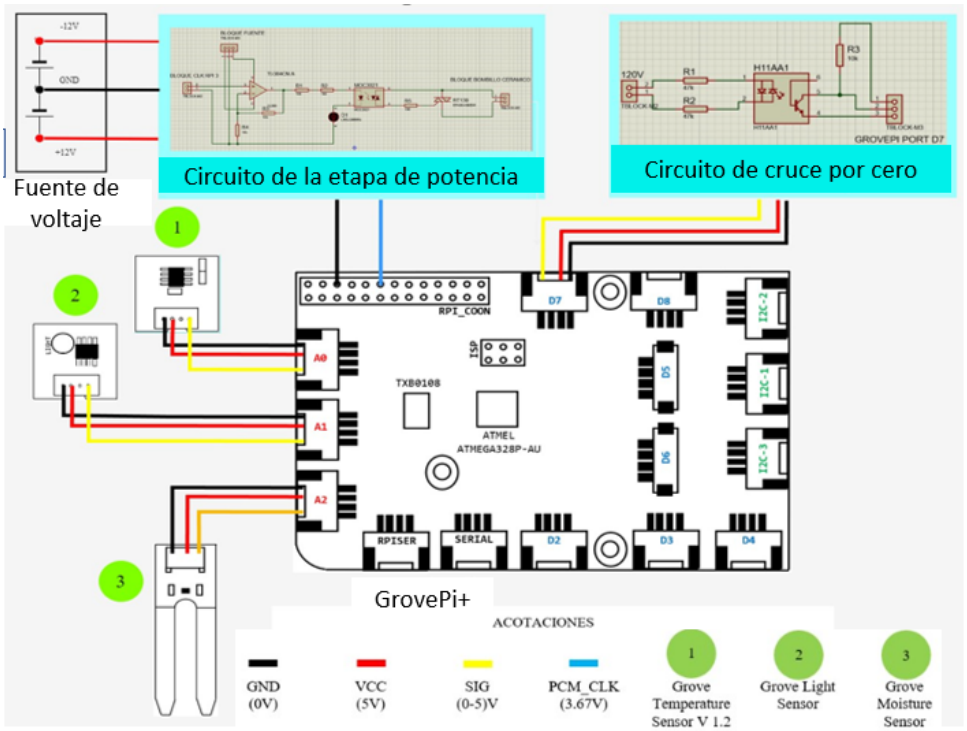


Figura 3.3: Sistema de monitoreo propuesto.

Los componentes de la Figura 3.3 forman parte de la base "A" del nivel 2 vista anteriormente en el modelo del invernadero. La Tabla 3.4 muestra los puertos, pines y elementos de los que depende cada tarea.

Adquisición de datos					
	Temperatura	**Luz**	**Humedad**	**Circuito de la etapa de potencia**	**Circuito detector de cruce por cero**
Puerto GrovePi+	A0	A1	A2	N.A	D7
Pin Raspberry Pi	N.A	N.A	N.A	6, 18 (BCM)	N.A
Componentes clave	Grove temperatura sensor	Grove light sensor	Grove soil moisture sensor	Fuentes de alimentación, Línea de 120V CA, resistores, Circuitos integrados.	Resistores, Circuito integrado.

Tabla 3.4: Tabla de adquisición de datos

3.3. Instalación de librerías necesarias para el desarrollo del sistema de monitoreo.

Este proyecto se basa en el uso de Raspberry Pi 3 B+ en el sistema operativo *Raspbian for Robots* o bien *Raspbian GNU/Linux 9 (stretch).*

Raspberry Pi debe ser acondicionada con los módulos y librerías necesarias para el desarrollo de las interfaces gráficas de usuario. Al utilizar el lenguaje de programación Python, el compilador de programas Thonny IDE y teniendo en cuenta el *hat* GrovePi+ se considera el uso de las siguientes librerías.

3.3.1. Grovepi.

Grovepi es una librería que ya viene instalada de manera predeterminada al utilizar el sistema operativo "*Raspbian for Robots*", por lo que, al indagar en la colección de scripts de muestra, esta librería se encontrara lista para manipular completamente los puertos de GrovePi+. La librería Grovepi tiene un gran alcance dado que es compatible con los módulos y sensores de la familia Grove y seeed studio.

3.3.2. Tkinter.

Esta es otra de las librerías que ya vienen instaladas de manera predeterminada dada la versión de Python instalada en el sistema operativo "*Raspbian for Robots*" por tanto para utilizarla se requiere de su importación desde el compilador Thonny IDE.

3.3.3. Matplotlib.

Esta es la primera librería a instalar, para ello se requiere de la terminal de comandos disponible y la instrucción:

sudo pip3 install matplotlib pytk

Al ejecutarlo se comienza la descarga de los paquetes, posteriormente se instalan y para poder confirmar el proceso se debe verificar que las barras de progreso concluyan al 100 %, cerciorarse que no haya surgido un error marcado en cada uno de los mensajes de finalización de subprocesos y mediante el compilador Thonny IDE realizar la importación de la librería ya sea con un programa de prueba o bien con el corrector de sintaxis.

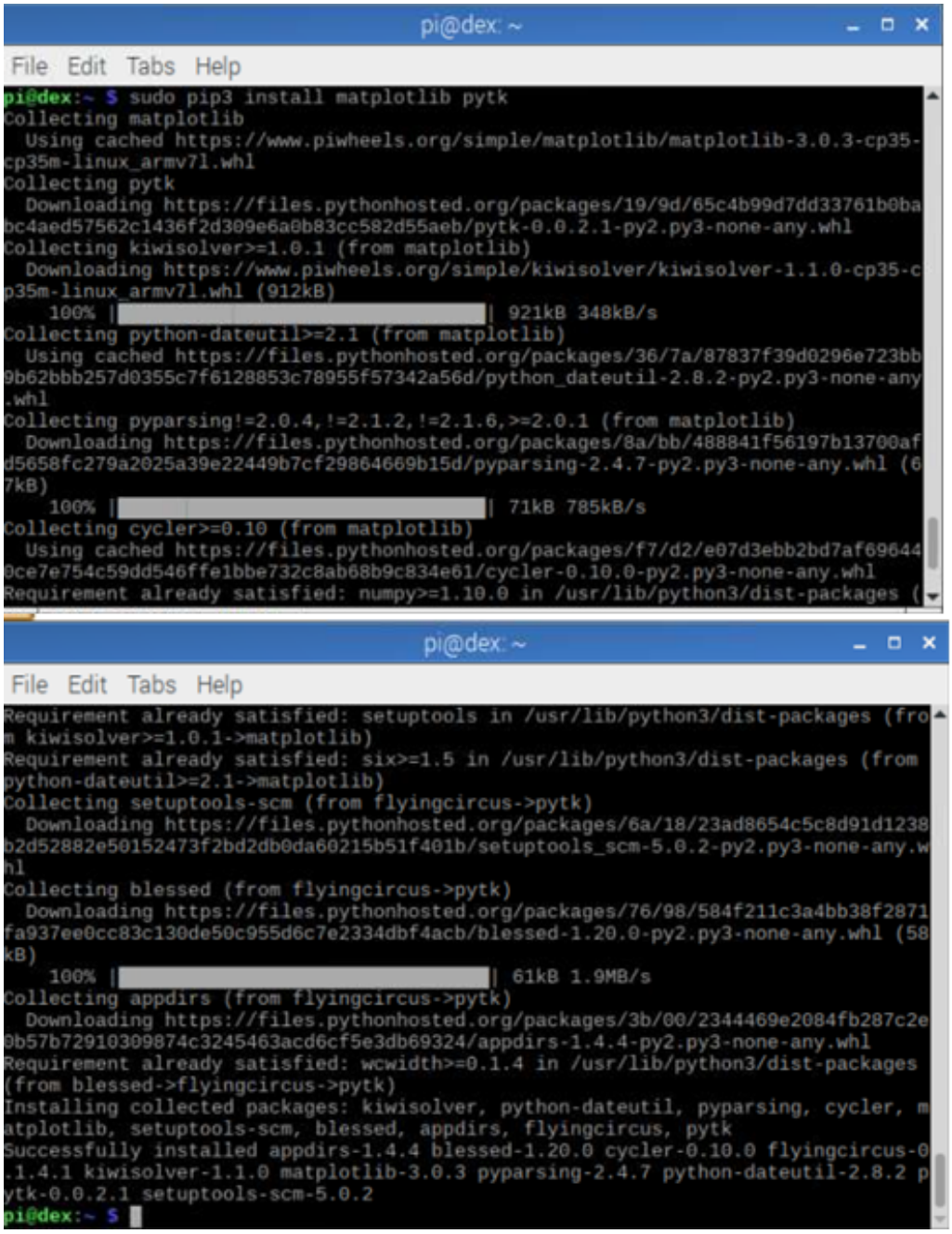


Figura 3.4: Proceso de instalación de la librería Matplotlib.

Al finalizar la descarga se deberá realizar la actualización de paquetes mediante dos comandos adicionales muy reconocidos en sistemas Linux/UNIX.

sudo apt update

Al terminar de descargar las actualizaciones se procede a realizar los cambios con el siguiente comando:

sudo apt upgrade

3.3.4. XlsxWriter

Para la instalación de la librería XlsxWriter se hace uso del siguiente comando:

pip install XlsxWriter

De igual forma se corrobora la instalación de manera correcta mediante la verificación de la barra de progreso, así como también de los mensajes de fin de subprocesos o bien como en este caso el menaje de fin de proceso ("*Succefully installed XlsxWriter-2.0.0*")

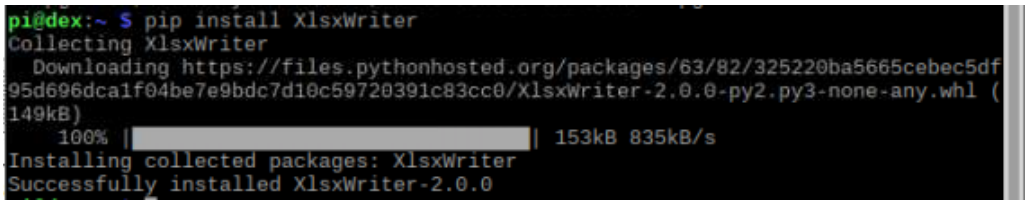


Figura 3.5: Proceso de instalación de la librería XlsxWriter.

Las librerías Grovepi, Tkinter, Matplotlib y XlsxWriter brindan las herramientas necesarias para la gestión de funciones del sistema de monitoreo.

3.4. Desarrollo de *scripts*.

Un sistema de monitoreo está conformado por conjunto de herramientas y/o tecnologías desarrolladas con el fin de supervisar el estado de los procesos en tiempo real. Se hace la recopilación de un conjunto de datos para un análisis posterior, para presentar información, que sirva de poyo en la toma de decisiones procurando optimizar la eficiencia y la detección de fallas. La observación de las variables del proceso es una de las funciones fundamentales en estos tipos de sistemas ya que sirven en gran medida en la emisión de alertas para generar seguridad y estabilidad.

En esta sección, se abordará de lleno el diseño de las interfaces que comprenden al sistema de monitoreo, se detalla con apoyo de los scripts del anexo A cada una de las interfaces. Se presenta a continuación los programas desarrollados desde el IDE de Thonny IDE, adquisición de lecturas de los sensores, codificación de las ventanas gráficas, elaboración del código del control PID así como del filtro de Kalman.

3.4.1. Adquisición de datos de las variables físicas a monitorear en el invernadero.

El código que hace posible la obtención de los valores de cada uno de los sensores está comprendido en un solo script o "clase", no es representado por medio de una interfaz ya que únicamente se trata de un módulo invocable en el programa principal cada vez que sea necesario. El código completo es presentado en el anexo A.1, y está estructurado de la siguiente manera:

En primera instancia se encuentra una función por cada sensor dando un total de 3 funciones, cada una de ellas está apuntando a un puerto analógico de la extensión GrovePi+; *lectura_hummodulo* está estrechamente relacionada con el sensor de humedad, *lectura_lummodulo* al sensor de luz y *lectura_temp* al sensor de temperatura. Las funciones requieren de un solo argumento que en este caso son las variables *moisture_sensor*, *light_sensor* y temp, en la Tabla 3.5 se puede llegar a observar el valor que adopta y su conexión con GrovePi+. El script está cargado en la ventana principal tomando el nombre de "*valor_sensores.py*", por lo que, al ejecutarse el programa principal (script del anexo A.2) los datos de los sensores se mostraran de manera inmediata en la interfaz principal del sistema de monitoreo.

Es importante recalcar que este módulo de adquisición de datos es imprescindible para el resto de funciones del sistema, en la interfaz del control PID, en la herramienta de grabado de datos y en la ventana de gráficos de las variables de temperatura, luz y humedad de suelo.

Argumento de la función	**Valor**	**Puerto relacionado**
temp	0	A0
light_sensor	1	A1
moisture_sensor	2	A2

Tabla 3.5: Argumentos de cada función con respecto al puerto de la GrovePi+

La manera en que pasaran los datos de cada sensor contiene la siguiente forma:

Variable= valor_sensores.funcion(pin_correspondiente)

En donde:

Función= *lectura_hummodulo* / *lectura_lummodulo* / *lectura_temp*

pin_correspondiente= *moisture_sensor* / *light_sensor* / *temp*

Para finalizar la descripción del código presentado, se adjunta el tipo de dato que retorna cada función, dado que los tres sensores son de tipo analógico la salida no es de la misma manera y eso se refleja en la Tabla 3.6.

Función	**Valor de retorno**
lectura_hummodulo()	Entero
lectura_lummodulo()	Entero
lectura_temp()	Flotante

Tabla 3.6: Tipo de valor asociado a las funciones de la adquisición de datos de los sensores.

3.4.2. Estructura del sistema de monitoreo

La estructura general para las interfaces gráficas del sistema de monitoreo que serán desarrolladas en Python se presenta en la Figura 3.6, de esta manera el usuario lograra comprenderlo e interactuar con él. En dicha Figura se puede observar de manera gráfica un total de 4 interfaces con las funciones que realizan.

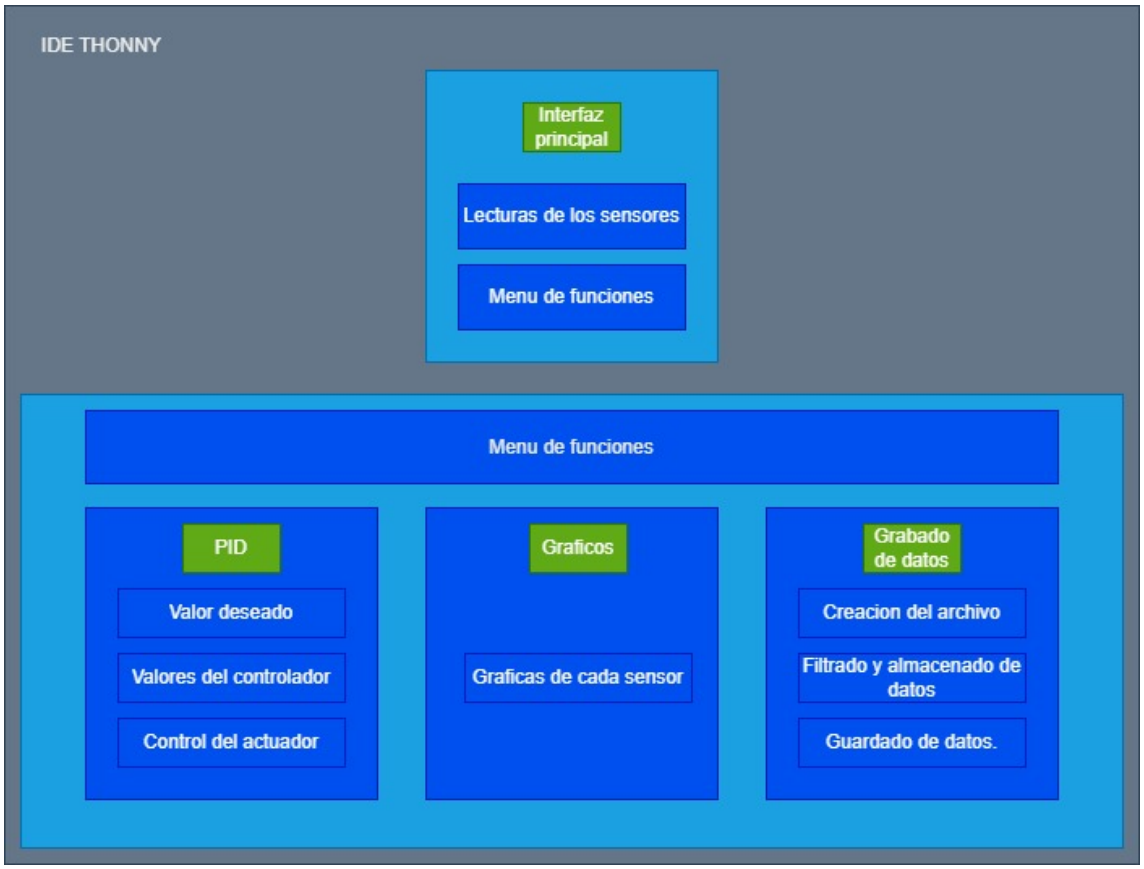


Figura 3.6: Representación abstracta de las interfaces gráficas del sistema de monitoreo.

La interfaz principal propuesta esta estructurada de tal manera que al usuario se le presenten los valores obtenidos de cada sensor además de un menú de 4 opciones que le permitan acceder a otras funciones.

Submenús del sistema:

1. PID. Para el control de la temperatura interna del invernadero.
2. Gráficos. Muestra 3 gráficas, cada una con la traza de una línea 2D del comportamiento de la temperatura, luz y humedad de suelo.
3. Grabado de datos. Para el almacenamiento de los datos correspondientes a la temperatura en formato de hoja de cálculo (archivo con extensión .xlsx).
4. Cerrar. Para finalizar con la ejecución del sistema de monitoreo.

Para la interfaz del control PID se contempla usar cajas de texto con el fin de poder digitar valores como:

- Valor de la temperatura deseada (setpoint).
- Valor de la ganancia proporcional.
- Valor de la ganancia integral.
- Valor de la ganancia derivativa.

Otro de los elementos a utilizar son los botones de acción, con el propósito de guardar los valores introducidos en las cajas de texto; un botón para iniciar el controlador así como uno más para detener la ejecución del mismo. El diseño de la interfaz se completará con la elaboración de etiquetas las cuales fungirán como indicadores del estatus del controlador, arrojando información como el error del PID, el ciclo de trabajo del pulso PWM, el error acumulado a través del tiempo, entre otros valores mostrados mas adelante.

Lograr la visualización de gráficos requiere de una interfaz o ventana principal, en donde, con la pulsación del submenú "Gráficos" localizada en la interfaz principal se desplegará una sub ventana dividida en 3 cuadrantes, y en cada uno de ellos se observará una gráfica 2D para cada variable física (temperatura, luz y humedad de suelo). Como tal esta función solo requiere de un botón, las gráficas se realizan con apoyo de Matplotlib.

La interfaz de grabado de datos contempla la librería XlsxWriter, esta función del sistema está contemplada para almacenar los datos obtenidos por el sensor de temperatura y que son indispensables para el análisis de sintonía del control PID. El filtro de Kalman estará presente en esta función dado que tomara muestras cada cierto tiempo y es necesario tener una medida lo más precisa posible. Para el diseño de la interfaz también se hará uso de cajas de texto ya que el usuario requiere de nombrar el archivo y necesitará de botones para iniciar el almacenamiento de datos y otro más para la finalización y guardado de la hoja de cálculo.

A continuación, se presenta el diagrama de flujo del programa que contiene el sistema de monitoreo, en el cual se hace énfasis a las distintas funciones y su lugar en el programa, es decir, en qué momento se pueden ejecutar y que es lo que realiza en sí.

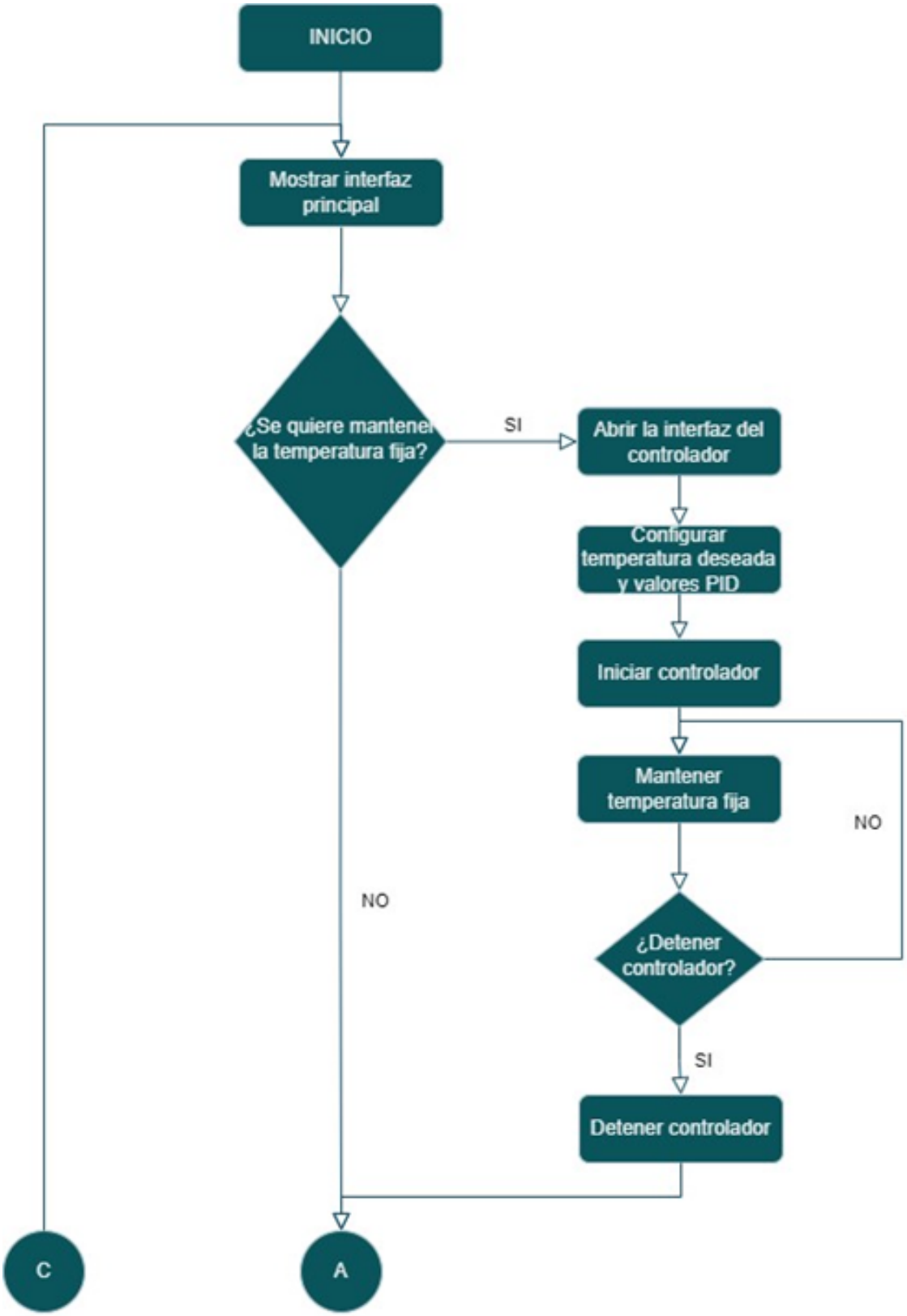


Figura 3.7: Diagrama de flujo del sistema de monitoreo parte 1.

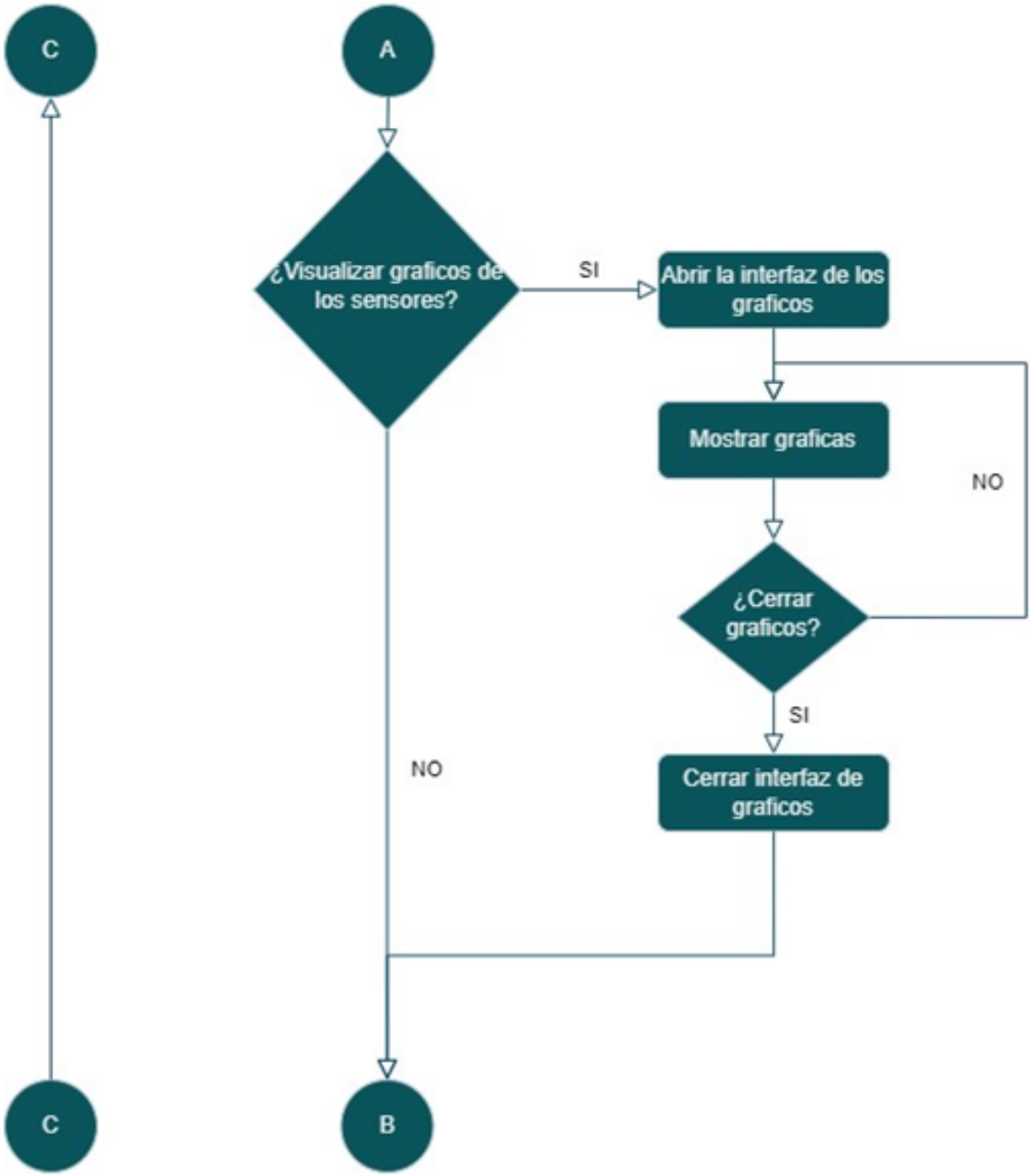


Figura 3.8: Diagrama de flujo del sistema de monitoreo parte 2.

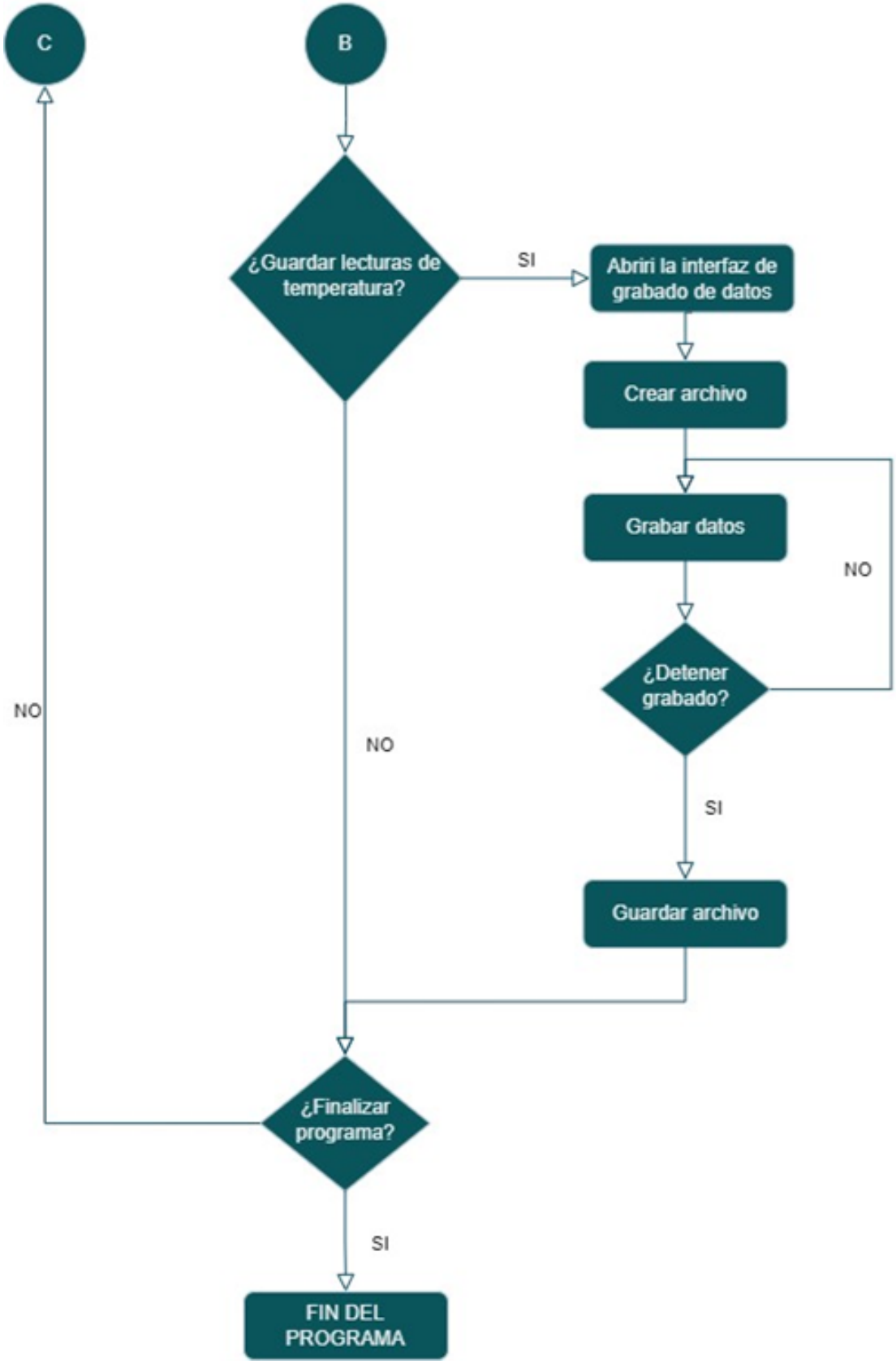


Figura 3.9: Diagrama de flujo del sistema de monitoreo parte 3.

3.4.3. Interfaz Principal

Tal como se observó en la Figura 3.6, está presente el diseño de una interfaz principal. Dicha interfaz da la bienvenida al usuario mostrando la información de cada una de las variables físicas además de un submenú en el que pueda acceder al resto de funciones descritas anteriormente.

El script desarrollado se encuentra en el anexo A.2, en él se pueden encontrar en la cabecera los subprogramas importados los cuales llevan por nombre "pid", "valor_sensores", "gráficos", "impresion_datos" y "grabado_datos". Cada uno de ellos será descrito con el fin de conocer su objetivo y la forma en que son programados (uso de funciones, métodos, cálculos necesarios, etc).

En las líneas 22 a 27 se realiza la configuración de pines mediante las instrucciones *grovepi.pinmode()* para determinar que puertos de la GrovePi+ se trabajaran como entradas o salidas.

Posteriormente, la creación de la ventana principal se lleva a cabo mediante el método *tk.Tk()* que directamente está contenida en la variable *root*. En esta última variable es posible personalizar el título de la ventana con el uso de la instrucción *root.title(" ")* recibiendo como argumento una cadena de caracteres.

Las dimensiones de la ventana son ajustadas con *root.geometry(" ")* recibiendo como argumento medidas en pixeles (ancho x largo). Para mostrar los valores de cada sensor se coloca la instrucción *app=impresion_ datos.App(root)* en A.2. En el momento en que se ejecute el programa interfaz_principal.py estará actualizando los valores por medio del subprograma *impresion_ datos.py*.

Para poder crear estos elementos se necesita asignar argumentos tales como la ventana en la que serán insertados (en este caso la variable root que identifica a la ventana principal), el texto que contendrá (campo *text*), el comando que se ejecutará al ser presionado (campo de *command*) sus dimensiones (argumentos *side* y *pady*) y el empaquetado por medio del método *.pack()*. Es importante aclarar que en caso de no empaquetar cada botón será imposible mostrarse en la interfaz y por ende tampoco se accederá a los gráficos, a la herramienta de grabado de datos y mucho menos al controlador.

De los comandos descritos, se puede observar en los botones *"PID"*, *"Graphs"* y *"Record Data"* una palabra reservada de la librería Tkinter (lambda), la cual es utilizada con frecuencia en el desarrollo de GUI's permitiendo llamar funciones ligadas al botón. Con estos elementos que se pueden ligar los subprogramas con la ventana principal, dando la oportunidad de implementar más funciones o bien, dicho de otra manera, haciendo escalable el sistema.

Al final del código se puede observar la instrucción *root.mainloop()*, encargada de ciclar infinitamente la interfaz. La manera de cerrar y parar el sistema se logra con el comando *root.destroy()* contenida en el botón denominado como *"Close"*.

3.4.4. Impresión de elementos de la ventana principal.

En el programa del anexo A.3 se tiene la finalidad de presentar la actualización de las lecturas de las variables físicas, su impresión en la ventana principal (interfaz_principal.py), fijación de iconos, y ubicar la rutina de encendido y apagado de las luces artificiales. Teniendo eso en cuenta, el programa interfaz_principal.py y el programa impresion_datos.py trabajan en conjunto para que la ventana principal del sistema de monitoreo logre presentar los datos al usuario tal y como se presenta en la Figura 3.10.

Los iconos de temperatura, luz y humedad de suelo mostrados en la Figura 3.10 no se integrarán en la versión final de la ventana principal y están allí con el propósito de presentar su ubicación y describir en la Tabla 3.7 la variable asociada en el programa mostrado en el anexo A.3. Es importante reiterar la función de empaquetado para los iconos en la ventana principal (*root*) y evitar tener problemas de visualización.

Variables asociadas a los iconos de las variables		
Identificador	**Variable**	**Posición**
1	**img1**	x=57, y=22
2	**img2**	x=149, y=22
3	**img3**	x=228, y=22

Tabla 3.7: Asociación de variables con los iconos de la ventana principal.

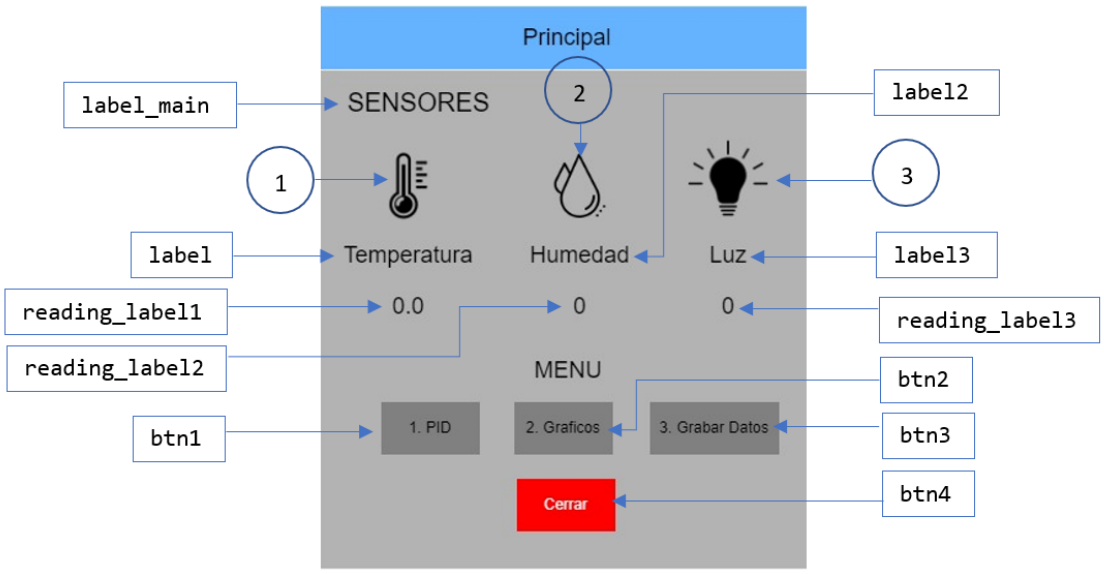


Figura 3.10: Asociación de variables y métodos para la creación de la interfaz principal.

De la interfaz presentada en la Figura 3.10 se espera tener la actualización de las mediciones de cada variable en un tiempo de 800mS de manera fluida sin representar problemas en el rendimiento de las otras funciones del sistema.
Una vez resumido las variables, métodos y funciones que comprenden el diseño de la ventana principal, es hora de ahondar en la función *update_reading()*; encargada de procesar las mediciones de los sensores y acondicionarla de tal forma que se impriman correctamente en la ventana principal.

La función *update_reading()* requiere de un argumento "*self*" para que, una vez procesadas las muestras permita realizar actualizaciones mientras se ejecuta el sistema de monitoreo, para ello es suficiente con observar las líneas 90-92 del anexo A.3 correspondientes a la lectura de luz. En estas líneas se realiza lo siguiente:

- En la línea 90 la variable *val1* toma la lectura actual de luz con apoyo del módulo de adquisición de lecturas (valor_sensores.py).
- En la línea 91 la variable *reading_str1* se encarga de hacer la conversión del número entero a una cadena de caracteres (*string*) dado que cualquier dato impreso en la interfaz debe ser de este tipo.
- En la línea 92 se realiza la actualización del valor de luz en la ventana principal.

Este procedimiento se repite para la variable de temperatura y la humedad. Basta con ver el bloque de código que está comprendido entre las líneas 89-102 de A.3.

3.4.5. Visualización de gráficos de las variables físicas.

La segunda función con la que cuenta el sistema de monitoreo es la visualización de gráficas 2D en tiempo real. Tal y como se presentó en el marco teórico, la librería Matplotlib hace posible la creación de gráficas personalizadas. Es necesario observar el código del anexo A.4 para comprender este apartado del documento.

En el código presentado en el anexo A.4 se presenta a la clase responsable del desarrollo de esta función del sistema. El nombre de la clase es "*graf*", y engloba la codificación del marco principal, su división para la creación de sub gráficas seccionadas por cuadrantes, funciones de captura de las mediciones, traza 2D para cada sub gráfica, así como las variables y parámetros necesarios para su implementación.

Este es el código en el que más librerías y sub programas se cargan, principalmente se importa la librería Matplotlib (módulos pyplot, animation y lines), en segunda instancia collections, seguido de time, grovepi y finalmente el sub programa valor_sensores. A continuación, se describen los elementos clave para su elaboración. El módulo *matplotlib.pyplot* permite crear gráficas y sub gráficas dentro de un mismo marco por lo que es un elemento fundamental en el programa. Con el uso de *matplotlib.lines* se logra el trazado de las líneas 2D y es gracias al método *matplotlib.animation* que se agrega dinamismo a cada sub gráfica dándole transiciones en cada intervalo de actualización.

Para el desarrollo del marco principal o "*main plot*" se recurre a la instrucción plt.figure() almacenada en la variable *fig* ubicada en la línea 26 (A.4). La construcción de los cuadrantes o "*sub plots*" se realiza con la extensión *add_ subplot*. De esta manera al marco principal se le agregan sub gráficas dentro de su espacio.

Para comprender de mejor manera, en el script se puede observar bloques de programación cuya sintaxis comienza con *axn*, donde *n* toma valores de 1 a 3 representando el número de sub gráfica que se está construyendo, por ejemplo, si se observan las líneas 27-32 se encuentra:

- En la línea 27 la creación del sub plot (o bien la posición del cuadrante al que corresponde el gráfico).
- En la línea 28 se determina el rango de valores para el eje X.
- En la línea 29 de acota el rango de valores para el eje Y.
- En la línea 30 el título principal que identificara a la sub gráfica (en este caso la temperatura).
- En las líneas 31-32 etiquetas para los ejes X y Y (X como tiempo y Y como el valor de temperatura).

De esta forma la distribución de las gráficas se posicionará de la siguiente manera:

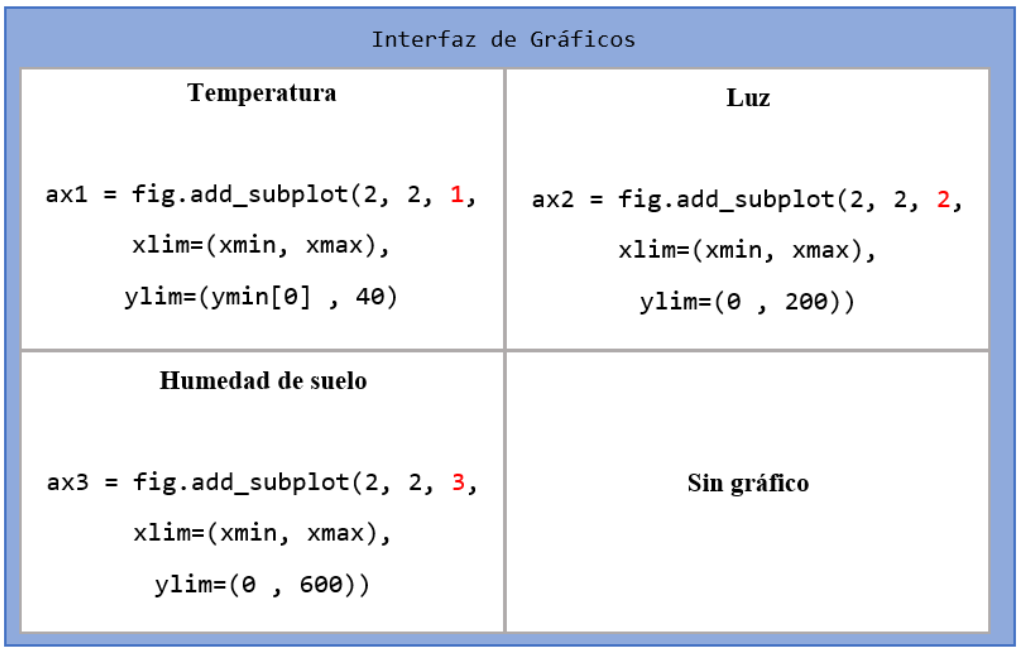


Figura 3.11: Distribución de sub gráficas en los distintos cuadrantes de la ventana.

Una vez creado el espacio de trabajo se requiere tomar los valores de cada variable y almacenarlos; para ello se hace uso de la función *CapturaDeValores*. Dentro de esta

función se obtienen dichos valores y se almacenan en un arreglo bidimensional 1x3 en el cual a cada posición le corresponde una variable, es decir:

Data = [ ]

	0	1	2
0	Temperatura	Luz	Humedad

El siguiente bloque del código está comprendido en un ciclo *for*, el cual lleva a cabo la tarea de agilizar la extracción de datos mediante la instrucción *collections.deque* además de trazar las líneas con *lines.append* cada vez que la instrucción de animación sea ejecutada.

La animación de las gráficas depende de *animation.FuncAnimation* (línea 73) cuyos argumentos deben contener; marco o "*plot*" principal (creado mediante la variable fig localizada en la línea 26), la función que provee los valores a gráficar (*CapturaDeValores*), las muestras por línea o trazo y el tiempo de actualización que tendrá.

Finalmente, se hace uso de la instrucción *plt.show()* para poder visualizar el gráfico completo. Es por esto que al momento de hacer clic en el botón "*Graphs*" desde la ventana principal (programa interfaz_principal.py) se desplegara el marco principal que contiene las sub gráficas de cada variable y por ende el trazado de las líneas según los valores que aportan los sensores.

3.4.6. Programa de la interfaz del controlador PID.

Para poder iniciar y observar el estado del controlador PID se desarrolló una pequeña interfaz, contemplando indicadores (o también llamadas etiquetas) que sirvan de ayuda en el monitoreo de la temperatura, así como los datos de lectura del sensor, el error, el valor de la suma del controlador PID y la conversión del PWM arrojado al actuador de calor. Ver script del anexo A.5 y A.6.

Esta sección se divide de la siguiente manera:

Figura 3.12: Control PID y filtro de Kalman en el sistema de monitoreo.

- Interfaz gráfica.

Al tratarse de una ventana emergente, al pulsar el botón "PID" en el programa interfaz_principal.py se apuntará a la clase del programa pid.py llamada APP2 encontrada en el script del anexo A.6. El argumento principal es "*root*" dado que la interfaz principal es la ventana de mayor nivel, y para poder generar ventanas "hijas" o sub ventanas es necesario ingresar en el programa *tk.Toplevel(root)*, instrucción que se almacena en la variable llamada *nw* (acrónimo de *new window*). Ya que esta ventana está diseñada para aceptar valores de sintonización digitados por el usuario, Tkinter provee los *TextBox* almacenados en las variables *entry*, *entry2*, *entry3* y *entry4*. Estas cajas de texto son útiles al momento de realizar pruebas de funcionamiento.

De manera gráfica se presenta el diseño de la ventana principal y los elementos que la conforman en la Figura 3.13 con el fin de resumir a grandes rasgos la estructura de la interfaz y facilitar su comprensión con respecto al script A.6. Brevemente se puede observar la forma en que se identifica cada elemento, es decir, para el caso de las cajas de texto se definen como entry, las etiquetas fijas como *label()* y los botones con las variable que los contienen (setpoint, proporcional, integral, derivativo, guardar y cerrar), los botones requieren de argumentos relacionados con las características de diseño tales como el tamaño, el nombre, el color de fondo, etc.

También se puede visualizar la presencia de diferentes etiquetas actualizables (*self*) que a diferencia de las etiquetas definidas como *label()* estas si pueden cambiar su valor inicial con el que fue codificado, esto es, que al iniciar con un valor de 0 al iniciar el controlador los valores se actualizan y las etiquetas con ellos. La actualización de estas etiquetas esta englobada en la función *capturar()* entre las líneas 224-229.

En el diseño de la interfaz se considero tanto la parte de la entrada de ganancias previo al accionamiento del controlador PID como el apartado de indicadores que apoyen en el monitoreo del controlador con el objetivo de visualizar la evolución de la temperatura en el invernadero.

Otra de las utilidades de los indicadores (Parámetros) radica en el tiempo de actualización del controlador, es decir, el usuario tiene la capacidad de modificar el tiempo de ejecución del PID conforme cambien los parámetros de manera que refine el control sobre la temperatura.

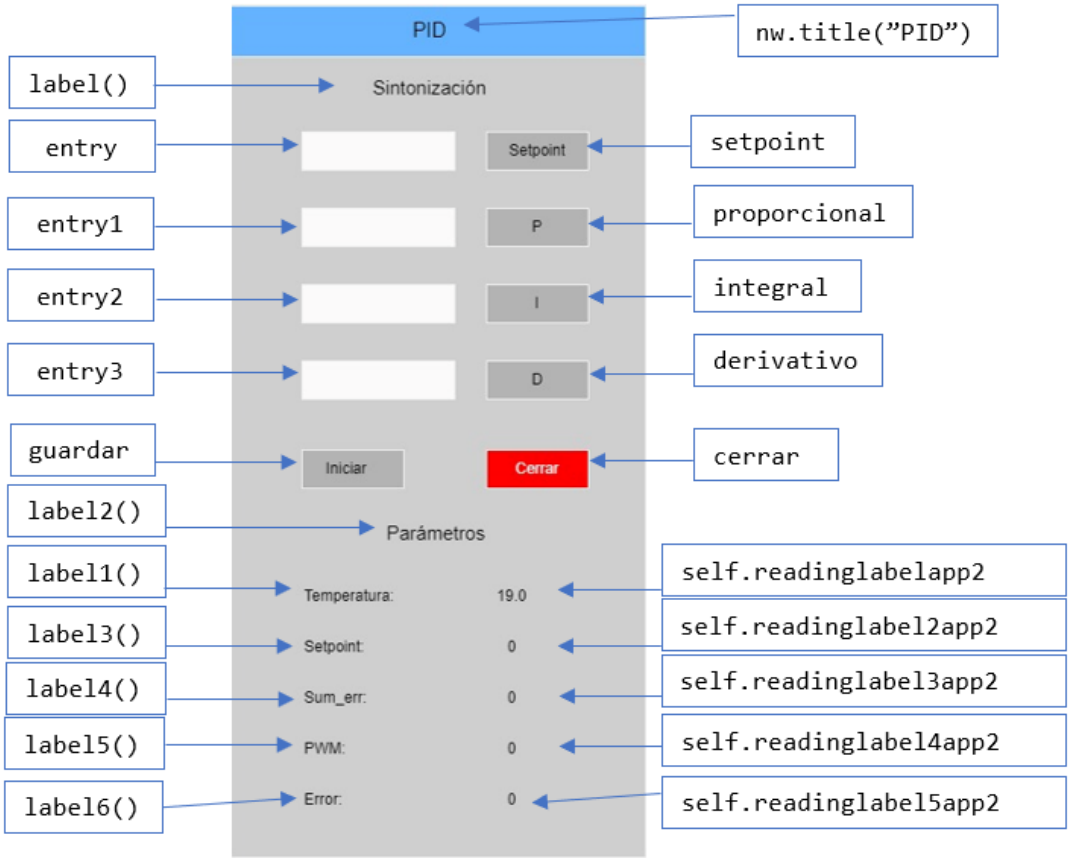


Figura 3.13: Asociación de variables y métodos para la creación de la interfaz PID.

3.4.7. Codificación del controlador PID y el filtro de Kalman.

Una vez descrito y mostrado el diseño de la interfaz que albergara el controlador PID de temperatura, en esta sección se hablara acerca de la implementación en la programación tanto del algoritmo de control como del filtro de Kalman. El script del PID se encuentra en el anexo A.5 y el código del filtro se encuentra en el cuerpo del programa de A.6.

En cuanto a las librerías a importar en el script de A.6 son; tkinter, grovepi y rpi.gpio. Los módulos o sub programas utilizados son únicamente 2; modulopid y valor_sensores. Este es el único programa que necesita de la librería rpi.gpio ya que, retomando la estructura del sistema de monitoreo presentada en la Figura 3.3 para poder comunicar a la Raspberry Pi con el circuito de la etapa de potencia del actuador se necesita de un pin en específico (pin 18 de la Raspberry Pi) que trabaje de acuerdo a un pulso PWM.

La configuración del pulso a través del pin 18 se realiza de la siguiente manera:

- Habilitar el uso de los pines GPIO y configurarlo con la numeración BCM (línea 20).

- Establecer el pin 18 como una salida digital (línea 23).
- Establecer la frecuencia de la señal PWM a una frecuencia de 120Hz (línea 24).
- Asociar el pulso PWM al pin 18 de Raspberry Pi (línea 26).
- Inicializar el pulso PWM en estado bajo antes de realizar cambios en su ciclo de trabajo (línea 27).

Con estas configuraciones iniciales ya es posible determinar distintos tiempos de trabajo en el pulso PWM de acuerdo a los cálculos realizados por el sub programa modulopid.py (anexo A.5).

El filtro de Kalman se desarrolla a partir de las ecuaciones propuestas por Beckar A. (2023) vistas en el marco teórico, y se colocan antes del calculo que realiza el controlador en la clase del anexo A.5. De esta manera se asegura que la lectura proveniente del sensor ingrese al controlador con el menor ruido posible.

Codificación del filtro de Kalman.

La programación del filtro debe estar contenido en el bloque de código de la interfaz del controlador PID. Una forma gráfica de verlo se muestra en la Figura 3.14 en donde, de las múltiples etapas que conforman la interfaz se describe unicamente el proceso de codificación del filtro. Gracias al gráfico se puede resumir la función del filtro en 6 etapas, que va desde la creación de las variables A, Q, R, X_n, P_n, la captura de la temperatura (linea 186), la ejecución de la etapa de predicción, el calculo de la innovación (diferencia del valor de la temperatura con respecto a la predicción realizada un paso atrás en la linea 192), la etapa de corrección que involucra el calculo de la ganancia de Kalman (linea 193), la temperatura optima (ecuación de corrección en la linea 194) y la actualización de la varianza (linea 195). Finalmente se pasa el valor optimo de la temperatura al modulo del controlador PID.

Es necesario ejecutar la interfaz del controlador, digitar las la temperatura deseada así como las ganancias P, I y D para ejecutar. El bloque del filtro es muy reducido, y esta comprendido entre las lineas 188-197 del programa del anexo A.6.

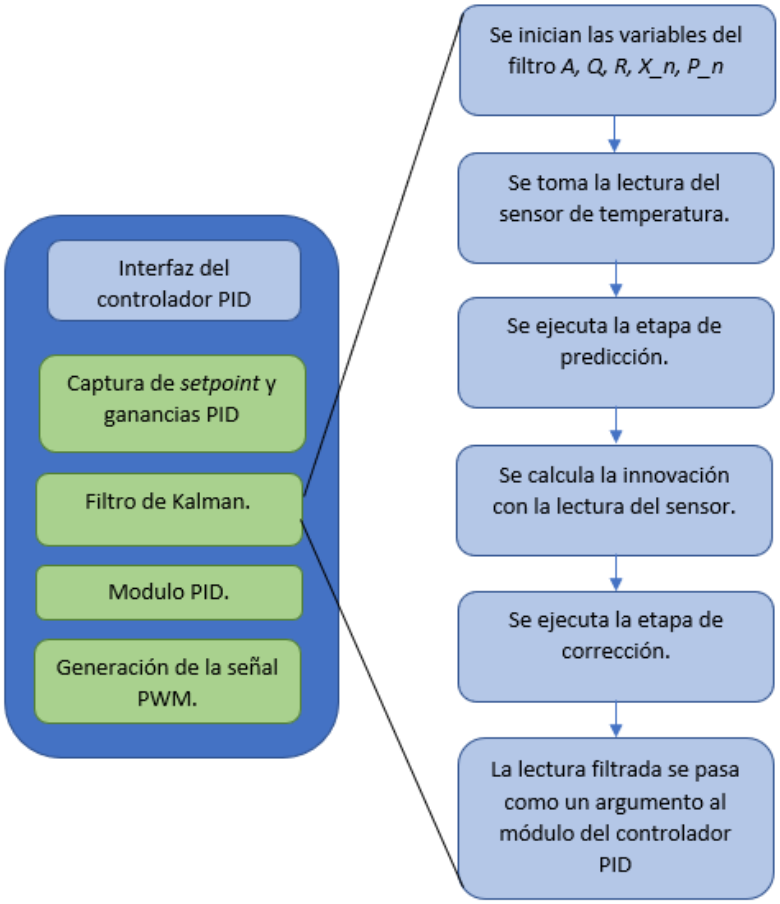


Figura 3.14: Proceso de codificación del filtro de Kalman.

En el código de A.6 se definen las variables para realizar los cálculos del filtro, se pueden encontrar en el encabezado entre las líneas 12 a 16. En la Tabla 3.8 se muestra el valor inicial que adopta cada una de ellas.

Variable en el programa	**Valor**
A	1
Q	0.01
R	1.0968
X_n	0
P_n	1

Tabla 3.8: Valor inicial de las variables del filtro de Kalman.

De acuerdo con las expresiones matemáticas que define el filtro de Kalman según [6], dado que el filtro solo tiene como variable la temperatura, se asume que el filtro sera de tipo unidimensional por lo que el cálculo de la predicción y corrección de los estados se simplifican de tal forma que las variables H y A (o también conocida <como la matriz F en la literatura) tendrán el valor de 1.

Las variables X_n y P_n descritas en las lineas 190 y 191 son colocadas en primer lugar ya que se requiere de ingresar los estados iniciales de la etapa de predicción, en el programa se asigna a X_n el valor de 0. En cuanto a la variable P_n de acuerdo con Simon [49] se coloca el valor de 0 si se conoce con precisión el modelo del proceso, en caso contrario puede adaptar algún otro valor. Un valor alto indica que no se tiene la certeza del proceso y el filtro convergerá mas rápido al estado real, para este caso se asume que la temperatura no cambia drásticamente por lo que el valor asignado sera de 1. De la Tabla 3.8 tanto la variable Q como R tienen asignados valores distintos, en el caso de Q (ruido del proceso) Simon [49] recomienda utilizar ruido de proceso ficticio para mejorar el rendimiento del filtro puesto que al implementarse suele violarse alguna de las reglas; si las características de ruido cambian por ejemplo (esto es que el ruido ya no sea de media cero, o que si se relacionen en distintos tiempos) por lo que se le asigna el valor de 0.01,mientras que para el caso de R (ruido de la medición) se determina con la varianza de las muestras tomadas de la curva de reacción de la temperatura interna del invernadero, por lo que se opta por obtener la varianza teórica de acuerdo a los datos experimentales guardados el cual es 1.0968. La instrucción que lo hace posible esta descrita en el programa del anexo B.1 linea 50 (v=$np.var(datv)$), en la cual toma los datos de temperatura almacenados en un arreglo para posteriormente guardarla en la variable "v".

Hasta ahora el programa del anexo B.1 y el programa A.6 tiene incorporado el filtro, sin embargo el programa del anexo B.1 no forma parte del sistema de monitoreo y unicamente se enfoca en apoyar con la localización de valores para la sintonización del PID, mientras que en el programa del anexo A.6 se realiza tanto el diseño de la interfaz, la integración del controlador digital y su etapa de filtrado dentro del sistema. El filtro de Kalman también esta presente en la herramienta de grabado de datos y esto es necesario ya que el control PID tiene un tiempo de actualización distinto a la herramienta de grabado de datos recordando que se toman muestras cada 10 segundos.

Módulo del controlador PID.

Una de las partes esenciales para la interfaz del controlador PID es el módulo que realiza el cálculo de valores para el controlador. El código asociado a este módulo se presenta en el anexo A.5, en donde se puede observar su estructura, la codificación consiste en una sola función que requiere de ciertos parámetros tales como:

- Valor de referencia denotado por la variable "*setpoint*".
- Valor de tipo flotante del control proporcional denotado como la variable "p".
- Valor de tipo flotante del control integral denotado como la variable "i".
- Valor de tipo flotante del control derivativo denotado como la variable "d".
- Valor de tipo flotante de la muestra filtrada del sensor de temperatura denotado como la variable "sf".

- Valor del error actual del controlador denotada por la variable "z".
- Valor del error anterior denotada por la variable "*erra*".

Es a partir de este conjunto de parámetros que se obtienen valores como el error actual, el error anterior, la suma del error y el valor del control PID acotado entre 0 y 1000.

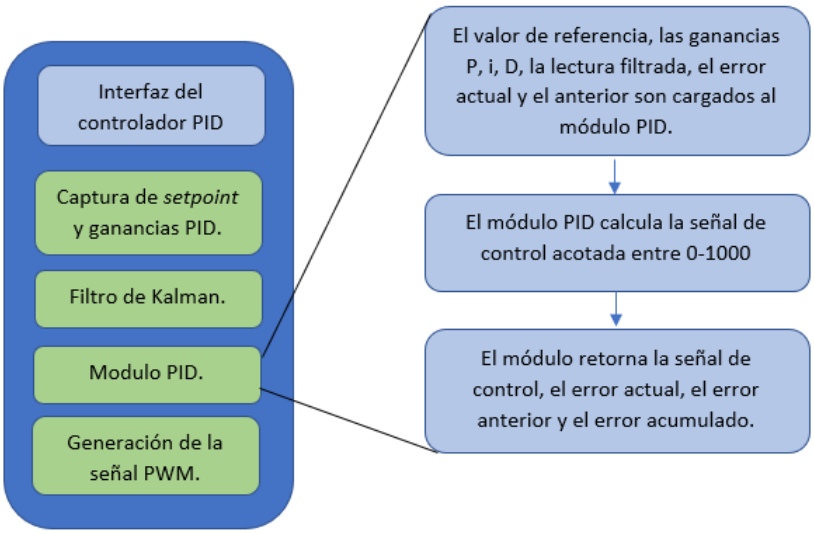


Figura 3.15: Proceso de codificación del modulo PID.

Los valores que retorna la función son 4, el valor actual del PID, el error actual, el valor del error acumulado y el error anterior como se muestra enla Figura 3.15. Esta información es constantemente actualizada siempre y cuando se inicie el controlador y se mantenga activa su respectiva interfaz. Este módulo es vital para el control en el suministro de energía al bombillo cerámico emisor de energía térmica. La formulación que se describe en la línea 7 de A.5 se considera a partir de la ecuación (2.1) que define a un controlador PID, como una interpretación para su implementación digital. De manera que, la señal de control está dada por la sumatoria de las 3 acciones de control; la ganancia proporcional multiplicada por la señal del error actual más la ganancia integral multiplicada por el error acumulado iteración a iteración más la ganancia derivativa multiplicada por la diferencia del error actual menos el error encontrado en la iteración anterior.

El valor de PWM se almacena en la variable j una vez pasa por una conversión de tipo de numero flotante a entero tal y como presenta en la línea 206 de A.6.

3.4.8. Sincronización del pulso PWM con la señal de voltaje de alimentación para el accionamiento del actuador.

Una de las tareas fundamentales es el control del suministro de energía al actuador por medio de la sincronización del pulso PWM para manipular el calor emitido

por el bombillo cerámico. Esto implica que la señal senoidal de la línea eléctrica de 120V AC debe iniciar en conjunto con el ciclo de trabajo de la señal PWM de manera que la potencia que reciba el bombillo cerámico emita calor acorde a la señal de control arrojado por el pin 18 de la Raspberry Pi.

En A.6 es gracias a la instrucción condicional de la línea 217 (*while*) de la mano con el circuito detector de cruce por cero que se evalúa si la señal de 120V AC llega a 0V AC, en caso de ser cierto la señal PWM es arrojada, procesada en el circuito de potencia y a su vez la energía eléctrica de la línea AC es cortada de acuerdo con el ciclo de trabajo de la señal PWM.

En la Figura 3.16 se presenta una simulación realizada con apoyo del programa LTspice para mostrar el comportamiento del circuito detector de cruce por cero, en ella se puede visualizar pequeños pulsos de 5V en los instantes en los que la señal senoidal de 120V pasa a 0V.

Los pulsos emitidos por el detector de cruce por cero entran al puerto digital número 7 del GrovePi+, por lo que la condicional while (línea 210 de A.6) entra en ejecución cuando se detecta el pulso.

```
while val==1:
        val=grovepi.digitalRead(salidacero)
```

En las líneas de código de A.6 se puede observar una instrucción de lectura al pin asociado con la variable "*salidacero*". Dicha variable arrojara únicamente dos valores ya sea 1 (en caso de recibir una señal de 5 volts) o 0 (en caso de no recibir voltaje en la entrada del puerto). Esta condicional no permitirá la ejecución del controlador PID y por ende tampoco el del filtro hasta no haber validado una señal de entrada en el puerto D7 para posteriormente hacer la sincronización del pulso PWM. Esta instrucción también puede llegar a ser útil para diagnosticar un problema con el circuito detector de cruce por cero, ya que si el controlador no responde se puede revisar el circuito en físico, o bien la entrada D7 del *hat* GrovePi+.

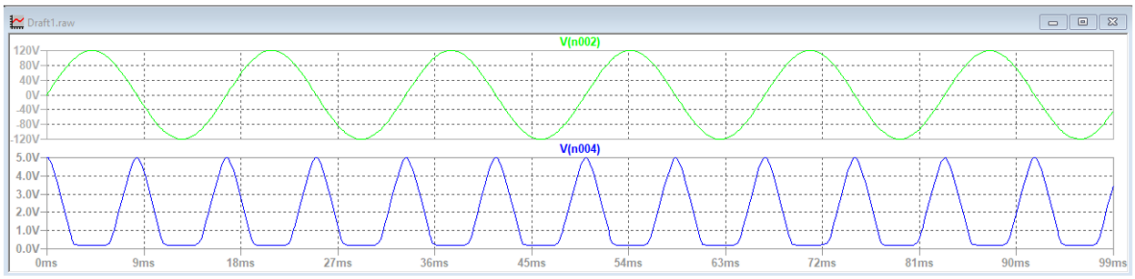


Figura 3.16: Simulación del circuito detector de cruce por cero.

3.4.9. Interfaz de captura de datos.

El código mostrado en el anexo A.7 se desarrolló con la finalidad de poder almacenar los datos de temperatura que provienen del sensor analógico *grove temperature*

sensor v1.2 ya que es necesario contar con el comportamiento del sistema por medio de datos cuantitativos que representen la curva de respuesta de la temperatura.

El módulo valor_sensores de nueva cuenta es fundamental, sin este bloque de código llevar a cabo la función de captura de datos no es posible. Es en este punto que se decide utilizar una librería adicional como lo es XlsxWriter que auxilia en gran medida en el almacenamiento de la información gracias a instancias y métodos que permiten trabajar de mejor manera con hojas de cálculo.

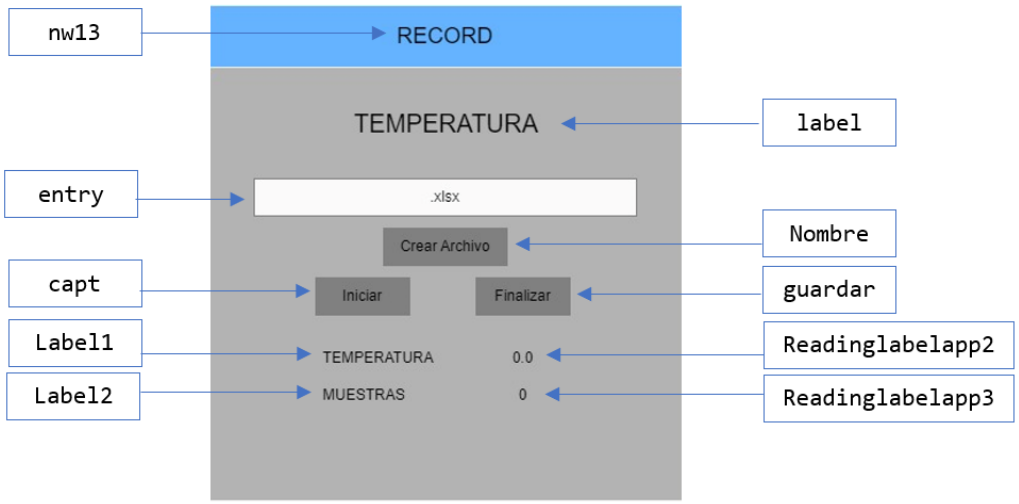


Figura 3.17: Asociación de variables y métodos para la creación de la interfaz de grabado de datos.

Ya que se requiere almacenar la lectura del sensor de temperatura, en el script se definen métodos para; creación de archivos, almacenado de información en celdas y detención de captura de datos. Al tratarse de una sub ventana, el método *Toplevel(root)* es necesario para poder crear la interfaz (línea 60), en dicha ventana se encontrarán alojados elementos como las cajas de texto o "*TextBox*", botones que como medio de acción de las funciones de creación de archivo, inicio y detención del almacenado de datos además de etiquetas que muestren la lectura grabada y el conteo de las muestras tomadas.

El método de creación de archivos se especifica desde la línea 40 con el método *guardar_nombre()*, en él se encuentran instancias pertenecientes a la librería XlsxWriter como lo son *xlsxwriter.Workbook*(línea 42) y *archive.add_worksheet*(línea 44). En donde, la línea 42 se encarga de crear un nuevo libro de trabajo y la línea 44 de la creación de la hoja de cálculo. Para que la herramienta pueda tener la facilidad de crear múltiples archivos sin la necesidad de detener la ejecución de la interfaz principal se opta por guardar los archivos creados en una lista (*cont_archivos=[]*), así, una vez que se detenga la grabación de datos inicial, baste con seleccionar la herramienta desde el menú principal de la interfaz principal, crear un nuevo archivo e iniciar una nueva grabación.

Es por medio del método *startdata()* (línea 142) que se inicia la grabación de datos, en primer lugar se guarda el valor de la temperatura en la variable "*val2*" así como la marca de tiempo en la variable "hora". Una vez almacenadas, para el caso de la temperatura se asigna el formato de tipo de dato flotante con 3 decimales de precisión. La lista definida como *cont_hojas* servirá para almacenar la hoja (perteneciente al libro de trabajo creado previamente) y poder utilizar la instrucción de captura con ayuda del método *write()*. Conforme se guardan las lecturas se actualizan las etiquetas con los datos mostrados en la ventana, cabe recalcar que el tiempo de muestreo es cada 10 segundos tomando en cuenta que la temperatura es una variable física que tarda en realizar cambio.

El método *stopdata()* como su nombre lo dice, tiene la finalidad de detener la grabación de forma correcta, para ello solo se necesita llamar el ultimo archivo almacenado en la lista *cont_archivos=[]* e invocar método *close()* (línea 47). Esta herramienta es útil ya que al contar con los archivos en formato xlsx permitirá más adelante utilizarlos para un análisis con ayuda de Matlab.

3.5. Circuito de la etapa de potencia.

Para entender de mejor manera la forma en que actuara el circuito de control de la etapa de potencia se presenta el siguiente diagrama:

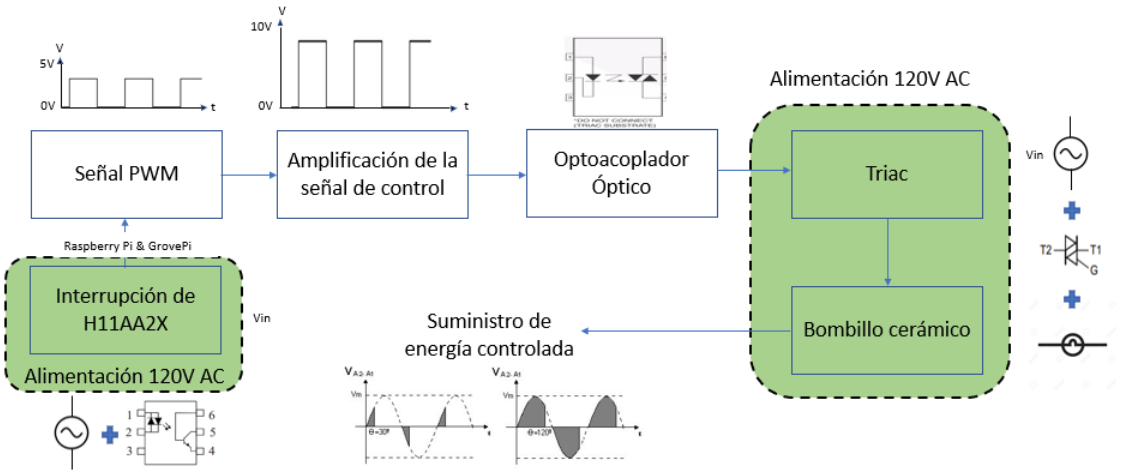


Figura 3.18: Panorama general del circuito de la etapa de potencia.

La tarea principal del circuito de control de la etapa de potencia es administrar la energía eléctrica que le llega al actuador. En la Figura 3.18 se puede observar de manera general el funcionamiento que tendrá una vez que el PID digital determine la señal de control PWM, todo comienza con una interrupción que el circuito integrado H11AA2X le envía a la Raspberry Pi, dicha interrupción le indica al software que debe lanzar la señal PWM con el ciclo de trabajo que el PID determino. Posteriormente se amplifica la señal de control para que el optoacoplador MOC3021 trabaje

en conjunto con el Triac BT139 y pueda administrar la energía al bombillo cerámico.

Inicialmente se tenía la propuesta de utilizar un pulso PWM proveniente del puerto digital 6 de la GrovePi+ sin embargo se descartó dado que no se logró obtener la frecuencia de la señal de salida deseada causando nulo control en el suministro de energía al bombillo cerámico. Por tanto, se optó por utilizar el pin número 18 de la Raspberry Pi (o también conocido como PCM CLK) ya que es posible manipular la frecuencia de la señal de reloj a 120Hz. Es importante destacar que, el pin 18 arroja el pulso con un voltaje de 3.44V estando al 100 % del ciclo de trabajo, esta cantidad es insuficiente para hacer accionar en su totalidad el optoacoplador MOC3021 con ciclos de trabajo inferiores, de allí que la amplificación por medio del operacional TL084CN es necesario. La configuración que se utiliza en el amplificador operacional es la de amplificador no inversor.

Diseño de los circuitos.

Una vez comprendido el funcionamiento de la etapa de control de potencia, a continuación se presentan las simulaciones realizadas en Proteus y LTspice así como el layout y la vista 3D para la fabricación del circuito físico.

En el diagrama de la Figura 3.19 realizado en Proteus, se observa la simulación realizada del circuito de la etapa de control de potencia. Para su implementación, se usará un bloque de terminal de 2 vías para las conexiones de los pines de PCM CLK y GND de Raspberry Pi, un segundo bloque (3 vías) destinado a las conexiones de la alimentación para el amplificador operacional por medio de las fuentes modelo s-60-12 y por último un tercer bloque (2 vías) para la conexión directa con el actuador (línea y neutro de la red eléctrica de 120V AC). La resistencia R6 es utilizada en la simulación para representar el bombillo cerámico en la implementación.

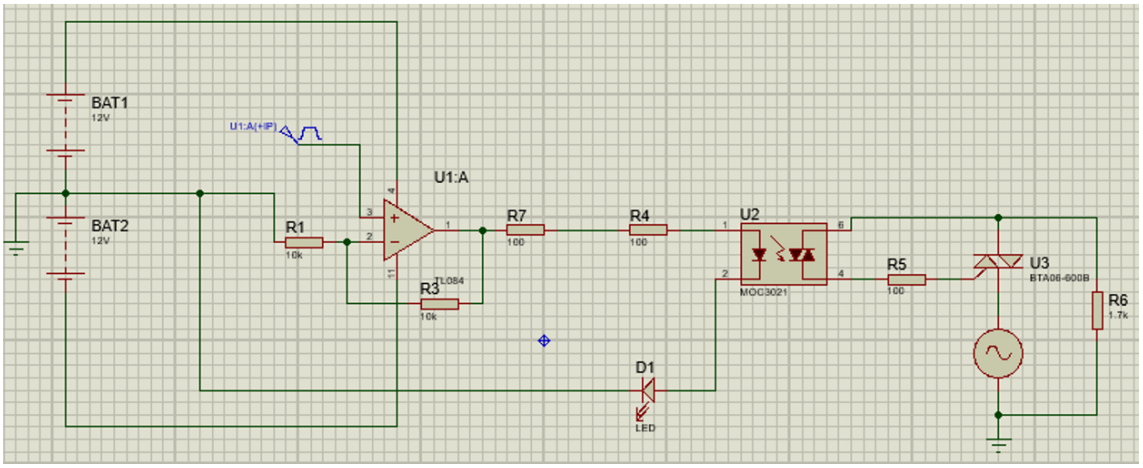


Figura 3.19: Diagrama de conexión para el circuito de potencia.

La señal PWM proveniente de la Raspberry Pi tendrá la siguiente ganancia después de ser amplificada según la ecuación 2.17:

$$A_{vf} = 1 + \frac{R_f}{R_1} = 1 + \frac{10k}{10k} = 2 \tag{3.1}$$

Por lo tanto, se espera tener un pulso PWM con un voltaje de 6.88V aproximadamente.

La señal resultante del circuito se presenta en la Figura 3.20, en la cual se muestran 3 simulaciones. Cada una de ellas muestra el comportamiento de la señal de 120V AC conforme cambia el ciclo de trabajo del pulso PWM para controlar el suministro de energía. Mientras más pequeño el ciclo menor energía es suministrada y viceversa.

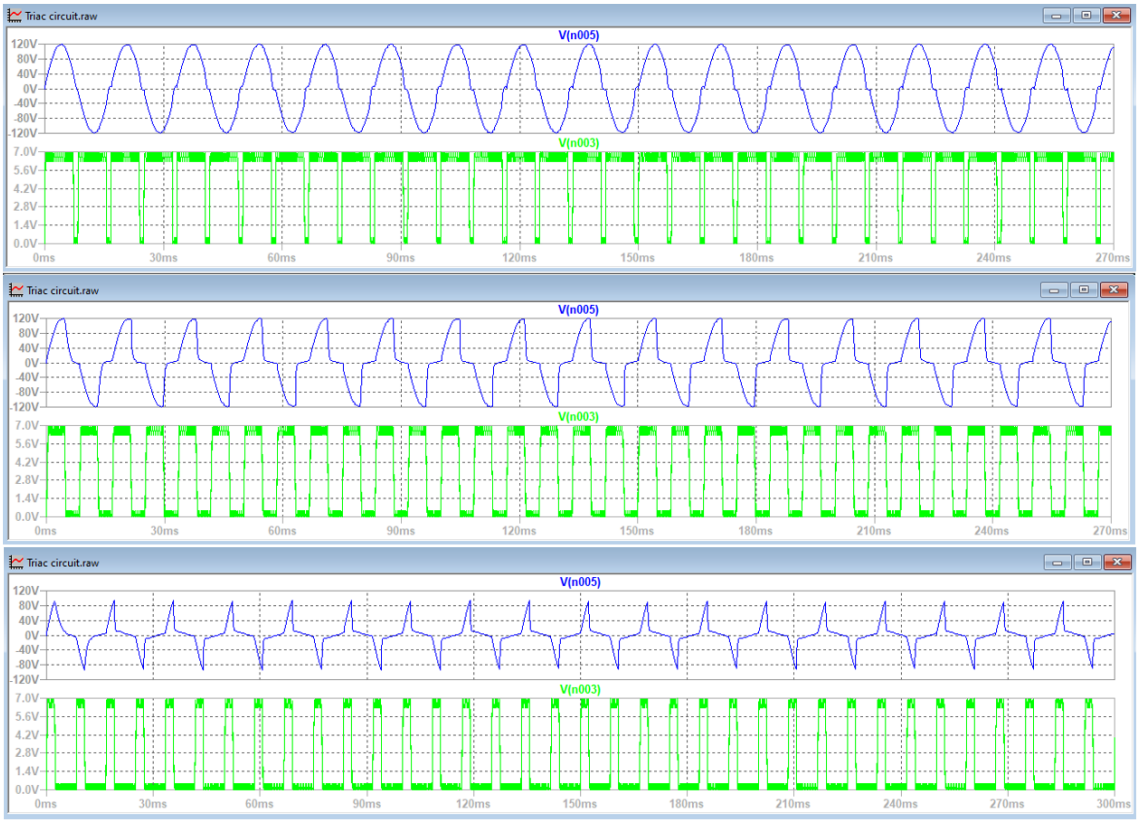
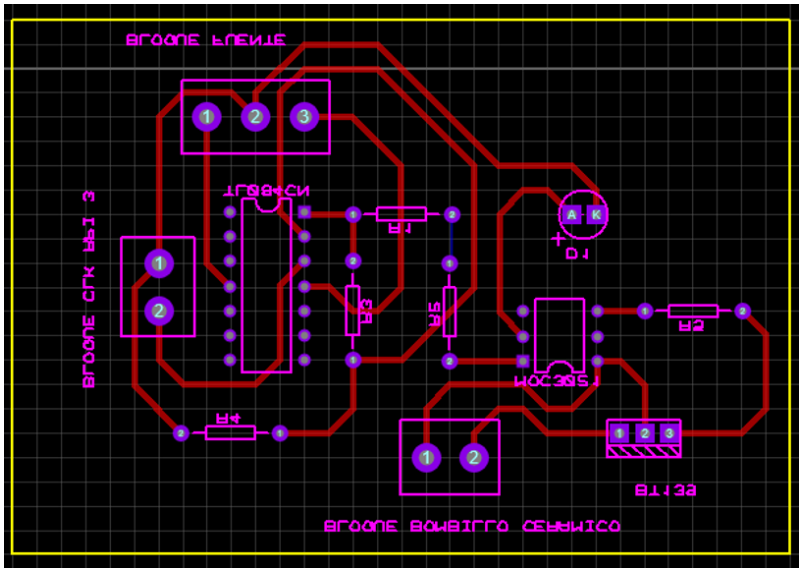
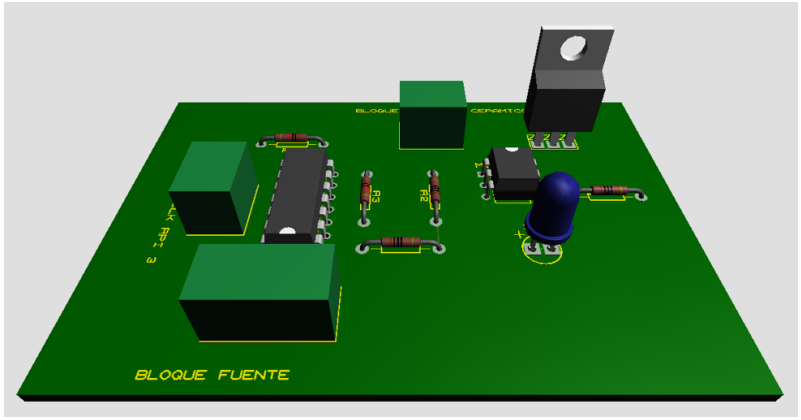


Figura 3.20: Simulación del circuito de potencia.

En la Figura 3.21a se puede observar el layout realizado para la PCB y en la Figura 3.21b el modelo 3D de referencia para su implementación. En estas figuras se presenta la distribución de los componentes electrónicos, optoacoplador, Triac, amplificador operacional, resistencias, los bloques de terminal, así como un diodo LED como indicador de funcionamiento.



(a) Layout del circuito.



(b) Modelo 3D.

Figura 3.21: Diseño del circuito de la etapa de potencia.

3.5.1. Detector de cruce por cero.

Este circuito tiene la finalidad de lograr sincronizar la señal de 120V, el circuito integrado principal es el H11AA1. Este C.I recibe como entrada una señal de corriente alterna (120V AC) teniendo como elementos intermedios resistencias de protección para los diodos del circuito integrado. A la salida se obtienen pulsos de 4.77 DC cada vez que se detecta el paso (o cruce) por cero en la señal de entrada. Una vez que se lanza el pulso, es por medio de la programación que es posible iniciar la señal de reloj proveniente de la Raspberry Pi a la misma frecuencia, con un voltaje de 3.44V DC y con distintos ciclos de trabajo (valores de 0 % a 100 %).

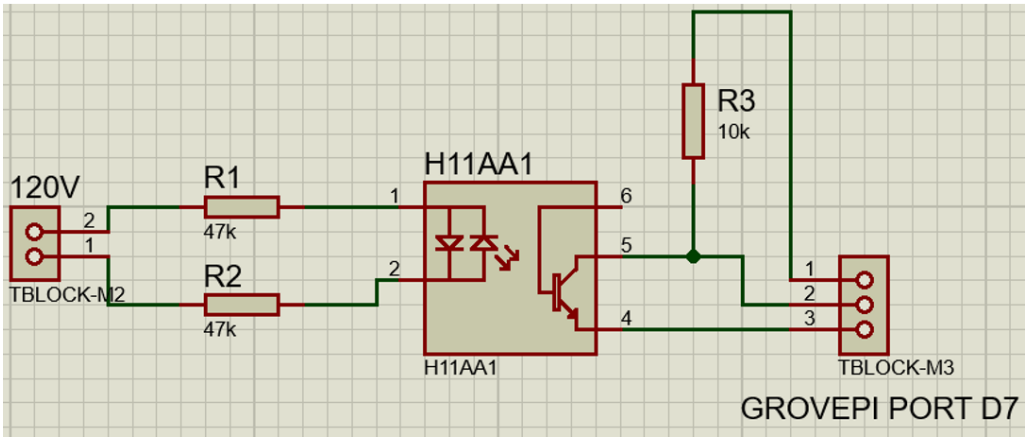
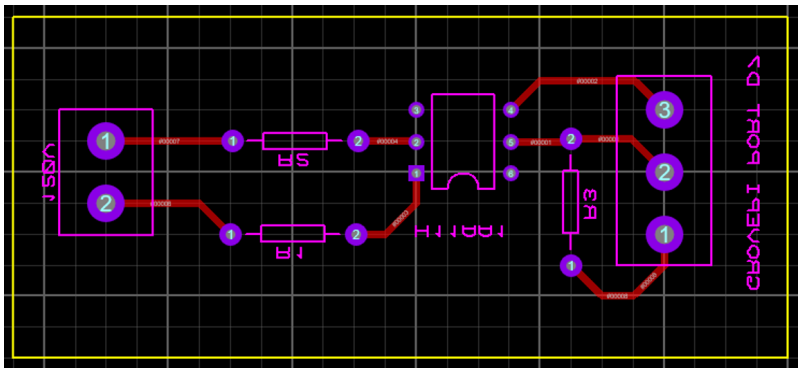


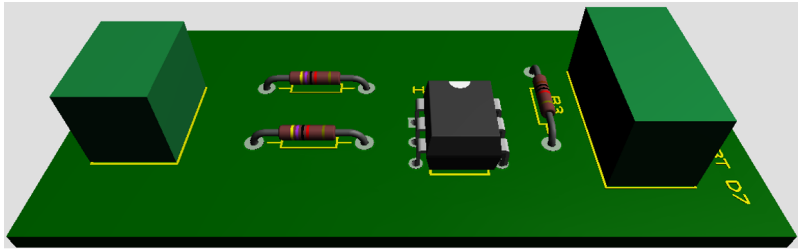
Figura 3.22: Diagrama de conexión para el circuito detector de cruce por cero.

Para la elaboración del circuito se realiza primero el diseño en el programa Proteus para posteriormente crear el modelo 3D e implementarlo en tablas fenólicas. El primer elemento se muestra en la Figura 3.23a, en la cual el layout muestra de manera simple las conexiones del circuito del detector de cruce por cero, se puede observar un módulo de 2 vías el cual recibirá la señal de 120V AC (conexión de línea y neutro) y un segundo módulo (3 vías) para la conexión con GrovePi+, las resistencias tienen un valor de 47K cada una.

En la Figura 3.23b se logra visualizar el modelo 3D del circuito que se espera llevar a cabo en las tablas fenólicas.



(a) Layout del circuito.



(b) Modelo 3D.

Figura 3.23: Diseño del circuito de la etapa de potencia.

3.6. Método de Ziegler-Nichols para el control PID de temperatura.

Dado que el método de Ziegler-Nichols es aplicable sin la necesidad de tener el modelo matemático de la planta, a continuación, se describe el proceso experimental necesario para determinar la ganancia proporcional, integral y derivativa partiendo desde la obtención de la curva de reacción de la temperatura. Posteriormente se realiza un análisis matemático para hallar los elementos necesarios (L y τ) que permitan encontrar el valor de las ganancias según las expresiones propuestas por Ziegler y Nichols y utilizarlas dentro del sistema de monitoreo en la interfaz del controlador.

3.6.1. Obtención de la curva de reacción de la temperatura interna del invernadero.

Para lograr obtener la curva de respuesta de la temperatura interna del invernadero se requiere de algunos pasos:

1. Tener conectado el sensor de temperatura *grove temperature sensor v1.2* al modulo de expansión/hat GrovePi+ en el puerto A0.
2. Ubicar el sensor de temperatura justo al centro del invernadero a 30 o 35 cm debajo del bombillo cerámico.
3. Tener preparada la interfaz de grabado de datos lista (recordando comentar del programa *grabado_ datos* el bloque del filtro de Kalman).
4. En la interfaz de grabado de datos crear el archivo en donde se almacenaran los datos de temperatura e iniciar la grabación.
5. Conectar el bombillo cerámico a la red eléctrica de 120V AC.
6. Esperar aproximadamente 2 horas.
7. Finalizar la grabación.
8. Comprobar que el archivo se encuentre en la misma ruta en donde se encuentra alojado el programa desde el gestor de archivos.

Una vez validado que el archivo se encuentre guardado y posea los datos de temperatura dentro de el, es necesario crear un archivo en la misma ruta llamado "FiltroKalman5"para que, en cuanto se utilice el programa del anexo B.1 no se encuentre con complicaciones al ejecutar. Es importante destacar que para el filtro de Kalman en este punto la varianza del ruido de medición no esta definida, es por ello que en el programa del anexo B.1 primero se coloco un valor pequeño (0.1) que sera reemplazado por la varianza que determine el programa después de su ejecución. En este punto se deberán tener ahora 2 archivos con extensión .xlsx, uno nombrado

FiltroKalman5 completamente vacío y otro con el nombre que el usuario le asigno en el proceso de grabación de datos.

Posteriormente se realizan los siguientes pasos:

1. Procurar mantener ambos archivos cerrados.
2. Desde Thonny IDE ejecutar el programa del anexo B.1.
3. Se deberá observar; 2 gráficas de la curva de reacción de la temperatura, una filtrada y otra no filtrada, las trazas verticales y las trazas horizontales para determinar los parámetros L y τ tal y como se presenta en la Figura 3.24.
4. Verificar en la consola de Thonny IDE la impresión del valor de la varianza de la curva de reacción.
5. Colocar en la variable R del programa del anexo B.1 el valor de la varianza impreso en la consola de Thonny IDE.
6. Volver a ejecutar el programa B.1

En la Figura 3.24 se puede encontrar un conjunto de lineas tanto verticales como horizontales las cuales sirven como guía para localizar la temperatura inicial y final de la curva de reacción (en el caso de las lineas horizontales), mientras que las verticales se trazan de acuerdo con la recta tangente para determinar los valores de τ y L fundamentales para calcular cada una de las ganancias del controlador PID con las expresiones propuestas por Ziegler y Nichols. La linea de la temperatura inicial y final se denotan en color verde tomando los valores mostrados en la Tabla 3.9, la recta tangente se gráfica con la ecuación 3.13 en donde, los puntos de intersección con las rectas horizontales ayudan a trazar las lineas verticales denotadas en color rojo.

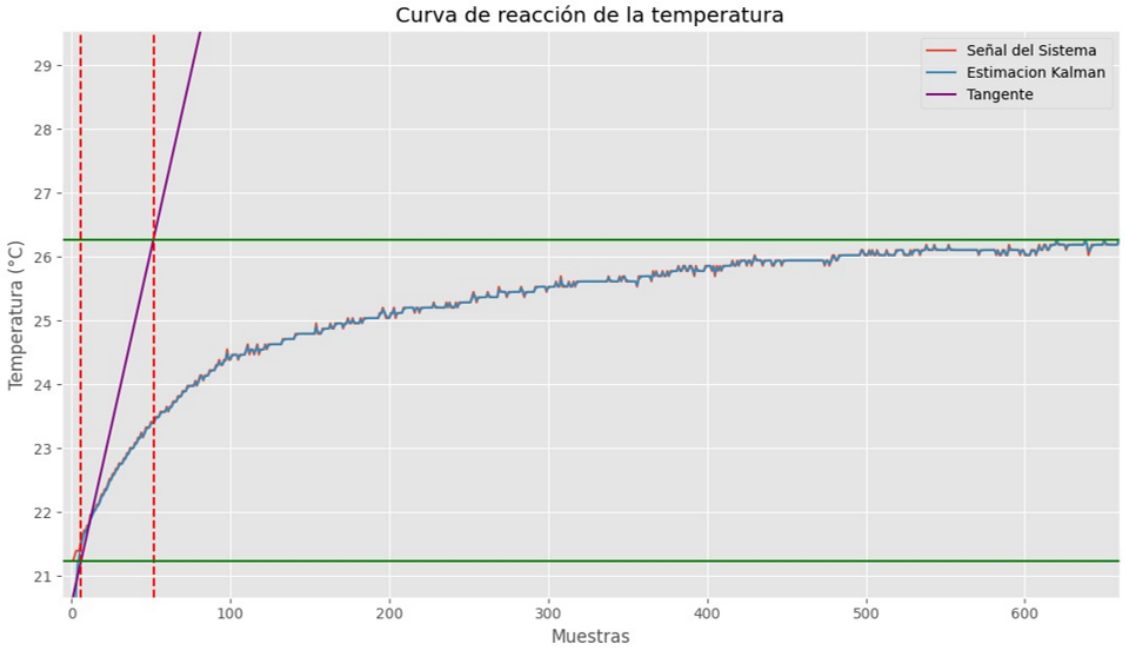


Figura 3.24: Gráfico de la temperatura interna del invernadero.

Haciendo hincapie en el programa presentado en el anexo B.1, se realizan las siguientes acciones:

- Se carga el archivo con extensión xlsx que contiene la información de la temperatura obtenida (linea 28).
- Se colocan las trazas horizontales de la temperatura inicial y final (lineas 85-88).
- Se realiza la traza de la recta tangente (linea 76).
- Se realiza el trazado de las rectas verticales que intersectan con la recta tangente (lineas 79-84).

Cada traza es fundamental para determinar el valor de L y τ.

De esta manera se obtiene la curva de reacción con y sin filtro de Kalman, a continuación se presenta la sintonización de los controladores PID.

3.6.2. Calculo de las ganancias del control PID sin el filtro de Kalman.

Una vez validado que las lecturas de la curva de reacción ya estén disponibles se toman algunos datos, entre ellos la temperatura inicial y la temperatura final como se muestra en la Tabla 3.9 y dos puntos como se puede observar en la Tabla 3.10.

Rango de temperatura	
Temperatura inicial	21.235°C
Temperatura final	26.273°C

Tabla 3.9: Valor inicial y final de la curva de reacción de la temperatura interna del invernadero.

De esos puntos se calcula la recta tangente y los valores L y τ.

- Obtención de la recta.

Puntos tomados de la gráfica	
Valores en x	**Valores en y**
x_1=**11**	y_1=**21.798°C**
x_2=12	y_2=**21.959°C**

Tabla 3.10: Identificación de puntos para la tangente de la recta tangente de la curva de reacción sin la etapa de filtrado.

Teniendo en consideración las ecuaciones de la recta y la pendiente:

$$y = mx + b \tag{3.2}$$

En donde la pendiente se determina como:

$$m = \frac{y_2 - y_1}{x_2 - x_1} \tag{3.3}$$

Para determinar el valor de b se resuelve la siguiente ecuación:

$$b = mx \tag{3.4}$$

Con base a esos puntos, se realizan los cálculos con ayuda de las ecuaciones 3.3, 3.4 para así determinar la ecuación de la recta tangente para el caso del PID sin el filtro de Kalman tal como se presenta a continuación:

$$y = 0{,}161x + 20{,}027 \tag{3.5}$$

Calculando a L y τ:

- Punto de intersección de la recta tangente con la temperatura inicial.

$$x = \frac{21{,}235 - 20{,}027}{0{,}161} = 7{,}5031 \tag{3.6}$$

El punto de intersección es el punto (7.5031,21.235).

- Punto de intersección de la recta tangente con la temperatura final.

$$x = \frac{26{,}273 - 20{,}027}{0{,}161} = 38{,}7950 \tag{3.7}$$

El punto de interseccion es el punto (38.7950,26.273).
El valor de τ para este caso es igual a:

$$\tau = 38{,}7950 - 7{,}5031 = 31{,}2919 \tag{3.8}$$

Mientras que L ya esta definido por la ecuación 3.6.

Una de las cosas a tener en cuenta es que, los valores obtenidos para τ y L están medidos en muestras, es decir, la forma de ver estos valores son como n numero de muestras, por lo que se debe realizar una conversión de tiempo. Para ello es importante recordar que los datos extraídos del archivo de excel contiene valores maestreados cada 10 segundos, por consecuencia, las conversiones se realizan de la siguiente manera:
Conversiones de tiempo.

$$\tau = 31{,}2919 muestras(\frac{10s}{1muestra})(\frac{1min.}{60s})(\frac{1hr.}{60min.}) \tag{3.9}$$

Por lo tanto el valor de τ es igual a 0.086921 hr.
mientras que para L:

$$L = 7{,}5031 muestras(\frac{10s}{1muestra})(\frac{1min.}{60s})(\frac{1hr.}{60min.}) = 0{,}020841hr. \tag{3.10}$$

Al calcular el coeficiente de controlabilidad se llega al siguiente resultado:

$$controlabilidad = \frac{L}{\tau} = 0{,}239 \tag{3.11}$$

A pesar de que el coeficiente de controlabilidad se encuentre dentro del rango se puede observar que sin la etapa de filtrado aumenta el coeficiente de controlabilidad y por ende mayor riesgo de perder el control del sistema.

Aplicando la Tabla de sintonización de valores PID de Ziegler-Nichols se determina:

Ganancia	Expresión matemática	Valor obtenido
Proporcional	$1{,}2\frac{\tau}{L}$	5.00480
Integral	2L	0.041682
Derivativa	0.5L	0.01042

Tabla 3.11: Calculo de ganancias sin el uso del filtro.

3.6.3. Cálculo de las ganancias del control PID implementando el filtro de Kalman.

Teniendo como referencia la gráfica de la Figura 3.24, las tareas que se llevan a cabo en el script del anexo B.1, se toman dos valores de la gráfica tal y como se presenta en la Tabla 3.9, Estos datos son relevantes en la búsqueda de la ecuación de la recta tangente.

De manera que la obtención de la recta utiliza los siguientes puntos:

Puntos tomados de la gráfica	
Valores en x	**Valores en y**
x_1=**11**	y_1=**21.78459°C**
x_2=12	y_2=**21.89512°C**

Tabla 3.12: Identificación de puntos para la tangente de la recta tangente de la curva de reacción con la etapa de filtrado.

Sustituyendo los puntos de la Tabla 3.12, el valor de la pendiente es:

$$m = \frac{21{,}89512 - 21{,}78459}{12 - 11} = 0{,}11053 \tag{3.12}$$

Tomando los valores de x_2 y y_2, el valor de m y b, la ecuación de la recta resulta ser:

$$y = 0{,}11053x + 20{,}56873 \tag{3.13}$$

Al tener definida la ecuación de la recta tangente, se ingresa en el script del anexo B.1 (linea 38), para obtener los puntos de intersección con la temperatura inicial y final el procedimiento es el siguiente:

- Punto de intersección de la recta tangente con la temperatura inicial.

$$x = \frac{21{,}235 - 20{,}56873}{0{,}11053} = 6{,}02774 \tag{3.14}$$

El punto de intersección es el punto (6.02774,21.235).

- Punto de intersección de la recta tangente con la temperatura final.

$$x = \frac{26{,}273 - 20{,}56873}{0{,}11053} = 51{,}607189 \tag{3.15}$$

El punto de intersección es el punto (51.607189,26.273). Mientras que para el valor de L se determina por la ecuación 3.14

$$\tau = 51{,}607189 - 6{,}02774 = 45{,}579449 \tag{3.16}$$

Mientras que para el valor de L se encuentra el valor de 6.02774.

Conversión del parámetro τ:

$$\tau = 45{,}579449 muestras(\frac{10s}{1muestra})(\frac{1min.}{60s})(\frac{1hr.}{60min.}) \tag{3.17}$$

Por lo tanto, de acuerdo a la ecuación 3.17 el valor de τ es 0.12660 hrs.
Conversión del parámetro L:

$$L = 6{,}02774 muestras(\frac{10s}{1muestra})(\frac{1min.}{60s})(\frac{1hr.}{60min.}) \tag{3.18}$$

Por lo tanto, de acuerdo a la ecuación 3.18 el valor de L es 0.016743 hrs.
Uno de los parámetros fundamentales en el control PID es el coeficiente de controlabilidad, definido por la razón L/τ acotada entre 0.1 y 0.3.

$$controlabilidad = \frac{L}{\tau} \tag{3.19}$$

Al sustituir los valores encontrados en las ecuaciones 3.17 y 3.18 en la ecuación 3.19 el coeficiente de controlabilidad es igual a 0.1322.

Dado que el coeficiente de controlabilidad se encuentra dentro del rango, el sistema de control es controlable.

Aplicando la tabla de sintonización de valores PID de Ziegler-Nichols se determina:

Ganancia	**Expresión matemática**	**Valor obtenido**
Proporcional	$1{,}2\frac{\tau}{L}$	9.0736
Integral	2L	0.033486
Derivativa	0.5L	0.0083715

Tabla 3.13: Calculo de ganancias con el uso del filtro de Kalman.

De acuerdo con los valores encontrados en la Tabla 3.13, ya es posible introducir cada ganancia en la interfaz del control PID del sistema de monitoreo para observar el comportamiento de la temperatura con distintos valores de referencia.

Capítulo 4

Resultados

Se presenta a continuación la construcción del prototipo del invernadero, la fabricación del circuito de control y el detector de cruce por cero en las tablas fenólicas, el diseño final de las interfaces, la interconexión de todos los elementos con Raspberry Pi y GrovePi+ y los resultados obtenidos de los controladores PID de acuerdo con las pruebas de funcionamiento realizadas.

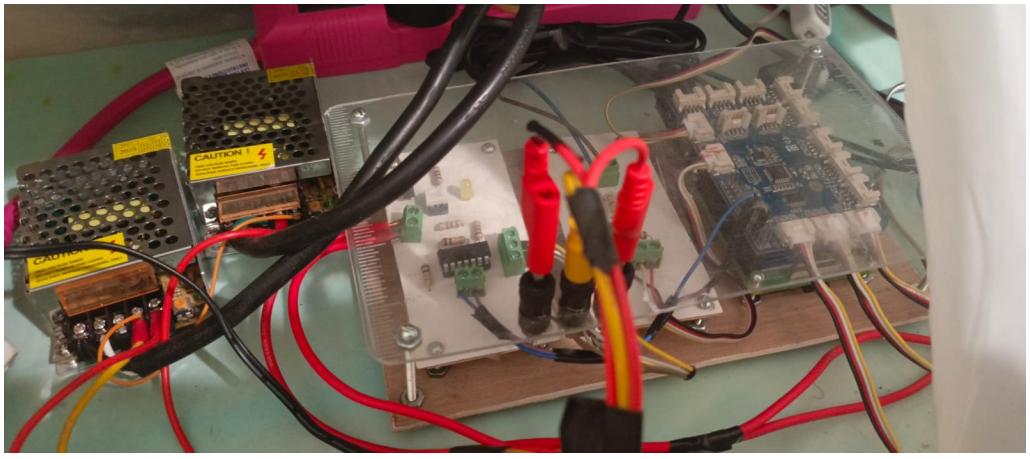
A lo largo del desarrollo de este proyecto, se llevaron a cabo diversas etapas con el propósito de alcanzar el objetivo general: el diseño e implementación de un filtro de Kalman para un controlador PID de temperatura acoplado a un sistema de monitoreo efectuado en un invernadero de pequeñas dimensiones. Estas etapas incluyeron desde el análisis e implementación del filtro de Kalman discreto para elaborar el algoritmo y aplicarlo al PID, la construcción del invernadero, la instrumentación de sensores grove (temperatura, humedad de suelo y luz), el desarrollo de interfaces gráficas de usuario y la sintonización del controlador PID por medio del método de Ziegler-Nichols. Cada etapa fue clave.

A continuación se presentan los resultados obtenidos en cada fase del trabajo, analizando el impacto del filtro de Kalman en la estabilidad del sistema de control y su efecto en la lectura del sensor de temperatura. Los resultados permitirán evaluar su estabilidad bajo distintas condiciones de operación.

4.1. Implementación del invernadero e instrumentación de sensores.

Para comenzar, de acuerdo con el prototipo del invernadero planteado en la Figura 3.1 a continuación se presenta el resultado de la implementación del invernadero, el cual sirvió como prototipo para poder realizar el ensamblado del sistema propuesto, se integra y conecta la tarjeta Raspberry Pi 3 B+ y GrovePi+. De la Figura 4.1a se pueden se encuentran presentes conectores Jack tipo banana atornillables colocados de manera que se puedan adaptar a las puntas provenientes de las fuentes de alimentación (+12V, -12V y GND).

Se instrumentan los sensores y se conectan al *hat* GrovePi+ en los puertos analógicos tal y como se presentó en la propuesta del sistema (Figura 3.3). En cuanto a la comunicación con Raspberry Pi 3 B+ es en software (programa del anexo A. 2 líneas 24-26) que se programa el puerto de conexión de cada uno de los sensores como una entrada para la adquisición de los datos.



(a) Interconexión del hardware.

(b) Vista exterior del invernadero.

(c) Vista interior del invernadero.

Figura 4.1: Implementación del sistema de monitoreo

4.2. Interfaces gráficas

En esta sección se presentan los resultados de la implementación del software del sistema de monitoreo de acuerdo al diseño de las interfaces gráficas mostrado en el capítulo 3. Las interfaces permiten visualizar en tiempo real las lecturas tomadas por cada uno de los sensores, mostrar de manera gráfica la evolución que tienen con el tiempo, etiquetas clave acerca de los cambios generados por el controlador PID e información de las muestras que se capturan con la herramienta de grabado de datos.

La ventana principal.

Primeramente, para la ejecución del sistema de monitoreo se deben seguir los siguientes pasos:

- Localizar el programa principal (anexo A.2) y abrirlo en el entorno de IDE Thonny IDE.
- Ejecutar el código con apoyo de la herramienta de compilación disponible en la barra de submenús.
- Verificar que la ventana principal del sistema muestre las lecturas de las variables físicas y los botones correspondientes a cada una de las funciones.

Al seguir los pasos, la ventana principal se muestra en la Figura 4.2. En ella se puede visualizar la temperatura en grados centígrados, el nivel de luz y el estado de la humedad de suelo, al usuario se le presentan 4 botones para acceder a las herramientas (PID, Gráficos, Grabar datos, Cerrar).

Figura 4.2: Ejecución del sistema del monitoreo (Ventana principal).

Gráficos 2D.

Una vez validado el funcionamiento de la ventana principal, para acceder a la función de los gráficos 2D únicamente se hace clic en el botón numero 2 e inmediatamente se logrará visualizar los 3 cuadrantes en una sub ventana tal como se presenta en la Figura 4.3. En dicha ilustración, se presenta en cada cuadrante el eje x representando el tiempo por medio de muestras (1-100), por otro lado el eje y varia el intervalo según la variable física, es decir, para el caso de la temperatura se acota de 0° C a 40° C, para la luminosidad y la humedad de 0 a 1000 dado que si es posible abarcar ese intervalo si el usuario cambia esas condiciones alterando el riego de las plantas o bien la intensidad de luz de las luces de crecimiento. Esta es una opción que el usuario tiene disponible a la hora de supervisar el comportamiento del control de la temperatura cuando se tiene activo el PID, la respuesta se mostrara constante de manera que el gráfico no se observaran picos o perturbación en la temperatura tal y como se presenta en la sub gráfica de la Figura 4.3.

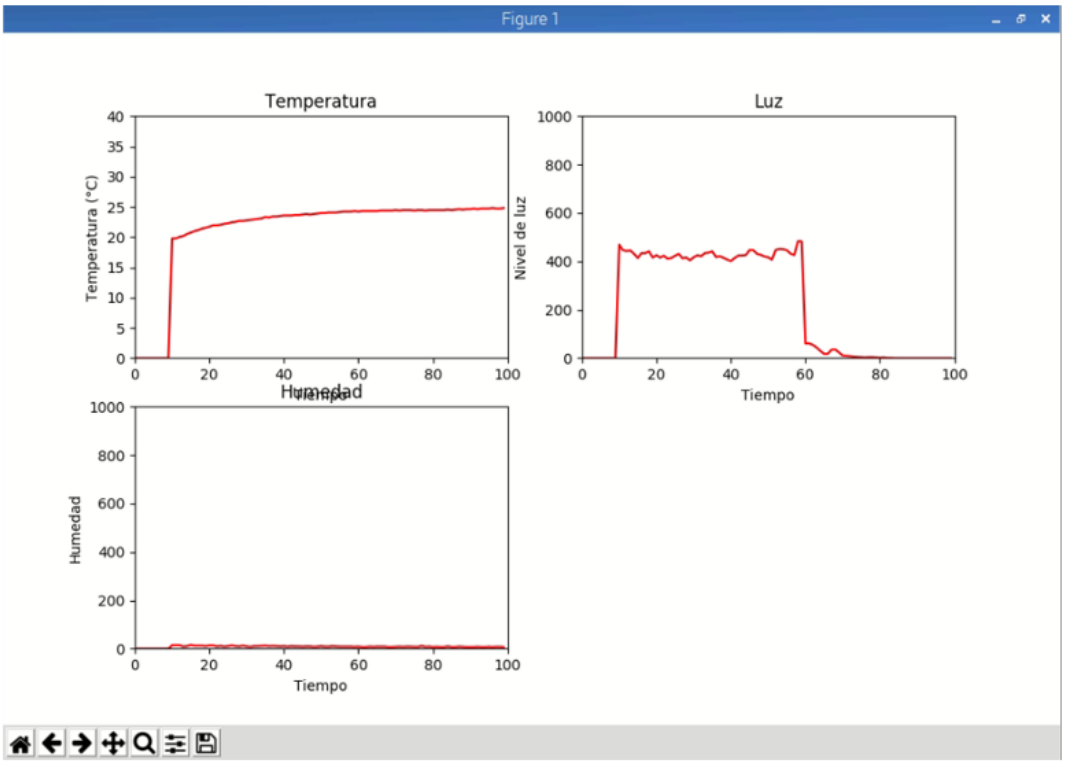


Figura 4.3: Ejecución del visor de gráficos.

Interfaz gráfica del controlador PID.

La interfaz del controlador se muestra en la Figura 4.4, en la cual debajo de la leyenda Sintonización se tienen agrupados los elementos fundamentales para configurar los parámetros del PID (setpoint, ganancia proporcional, ganancia integral y la ganancia derivativa). Por otro lado, debajo de la leyenda Parámetros se encuentran etiquetas las cuales muestran la información con respecto al controlador PID

cuando este entra en ejecución, tales como el setpoint ajustado por el usuario, el valor actual del controlador (sum_err), el valor de PWM y el error actual-

Configuración del PID de temperatura.

En cada caja de texto el usuario digital:

- La temperatura deseada.
- El valor de la ganancia proporcional.
- El valor de la ganancia integral.
- El valor de la ganancia derivativa.

Una vez digitado cada uno de ellos, es necesario pulsar los botones para guardar la información en el bloque de código del controlador. El PID entra en ejecución posteriormente al pulsar el botón iniciar. La manera de comprobar su ejecución y funcionamiento es por medio de la actualización de las etiquetas ubicadas debajo de la leyenda Parámetros.

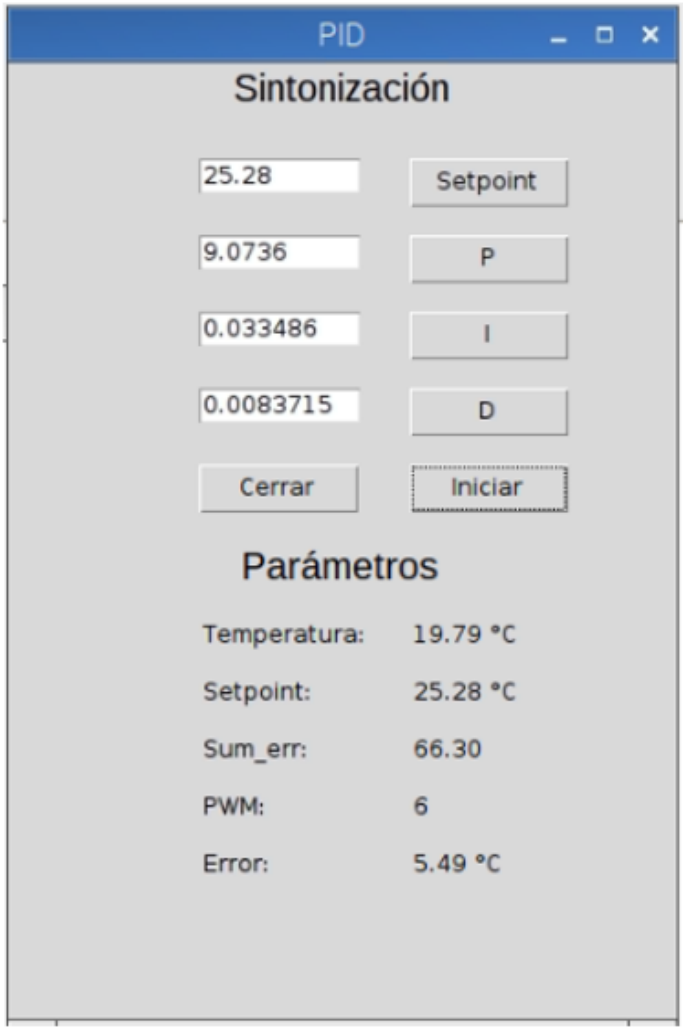


Figura 4.4: Ejecución de la interfaz del control PID.

Para detener la ejecución del PID de temperatura únicamente se pulsa el botón de cerrar, además de verificar que el led del circuito de la etapa de potencia se apague completamente.

La interfaz de grabado de datos.

Por último, la interfaz de grabado de datos se presenta en la Figura 4.5a, en la cual se muestra el resultado final de su implantación. En ella principalmente el usuario tiene la posibilidad de nombrar, crear archivos y almacenar datos de temperatura en ellos. Adicionalmente se tienen dos etiquetas "Temperatura" y "Muestras" las cuales le muestran al usuario la cantidad de muestras que se han guardado en el archivo y el valor que se ha almacenado de igual manera.

Configuración.

- Pulsar el tercer sub menú de la ventana principal "Grabado de datos".
- Nombrar el archivo antes de crearlo.
- Crear el archivo pulsando el botón "Crear Archivo".
- Inicializar el almacenado de datos en el archivo pulsando el botón "Iniciar".
- Verificar el proceso de almacenamiento de datos por medio de la actualización de las etiquetas de "TEMPERATURA" y "MUESTRAS".

Finalmente, para detener la grabación de datos se pulsa el botón "Finalizar", esta acción cierra el archivo creado y se guarda en la misma ruta en donde se localiza el programa del sistema de monitoreo. La forma de asegurarse de que realmente se ha guardado correctamente el archivo es por medio de una subventana con la leyenda "DATOS GUARDADOS" (Figura 4.5b). Los datos de temperatura se guardan en una sola columna del archivo excel.

(a) Ventana de creación de archivo, inicio y detención de la grabación.

(b) Interfaz de verificación de guardado del archivo.

Figura 4.5: Ejecución de la interfaz de grabado de datos.

Las interfaces gráficas desarrolladas facilitaron significativamente la interacción del usuario con el sistema de control de temperatura permitiendo una configuración intuitiva del PID y una visualización clara de los datos en tiempo real. El filtro de Kalman logra trabajar a la par con el control PID sin generar congelamiento entre interfaces que altere el rendimiento del sistema de monitoreo. En la Figura 4.6 se puede apreciar el resultado de ejecutar todas las interfaces del sistema de monitoreo, en donde es importante recalcar que el uso del CPU es bajo (9 %) pese a estar ejecutando la interfaz principal, la herramienta de gráficos 2D, el controlador PID (con el uso del filtro) y el constante almacenamiento de los datos de temperatura en la herramienta de grabación. Por lo que el uso de Raspberry Pi permitió adaptar el sistema propuesto tanto en el hardware como en el software.

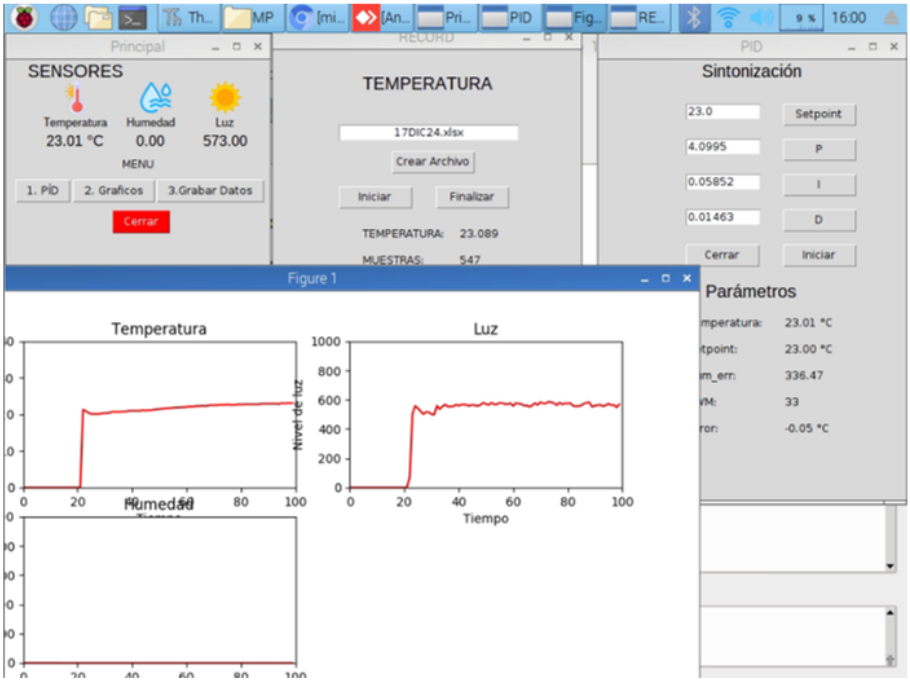


Figura 4.6: Ejecución de todas las interfaces del sistema de monitoreo.

4.3. Implementación de circuitos.

Con el fin de llevar a cabo la implementación del hardware del sistema propuesto, se presenta a continuación el resultado de la fabricación de los circuitos tanto de la etapa de potencia necesario para el control en el suministro de energía eléctrica del bombillo cerámico, así como del circuito de cruce por cero. Se valida el funcionamiento por medio del monitoreo de las señales de salida de cada uno de ellos con apoyo del osciloscopio digital.

4.3.1. Circuito de cruce por cero.

Para el caso del circuito detector de cruce por cero, se consideran los siguientes materiales y componentes en base al circuito mostrado en la Figura 3.3.

- 2 resistencias de 47KΩ.
- 1 resistencia de 10KΩ.
- Optoacoplador H11AA2X.
- 1 bornera de 2 vias.
- 1 bornera de 3 vias.

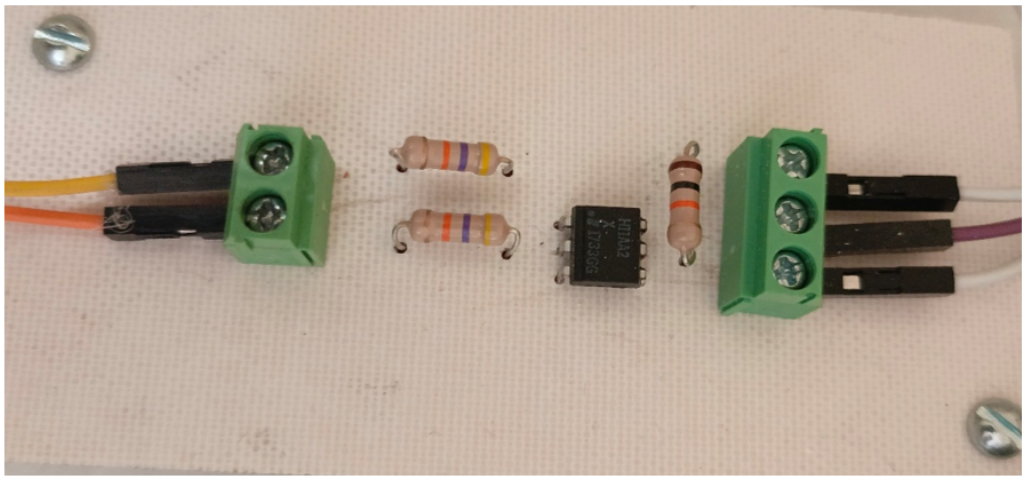


Figura 4.7: Elaboración del circuito detector de cruce por cero.

De acuerdo al circuito elaborado con el software Proteus anteriormente, se presenta en la Figura 4.7 el resultado de su fabricación. En ella se observan cables (jumper) conectados a cada una de las terminales de las borneras, este tipo de cable es utilizado en primera instancia para poder llevar a cabo pruebas de funcionamiento antes de integrarse a la Raspberry Pi y Grove Pi+, al menos para la conexión de la red eléctrica de 120V AC. Se realiza el cambio por cable de cobre calibre 18 para evitar daños.

Para evaluar su comportamiento se apoya de un osciloscopio que permita visualizar la señal de salida del circuito. Al conectar la línea de 120V AC en la bornera de 2 vías, el circuito integrado H11AA2X da como resultado la señal de la Figura 4.8 en la cual se pueden percibir pulsos de 4.72V DC a una frecuencia de 120.2Hz.

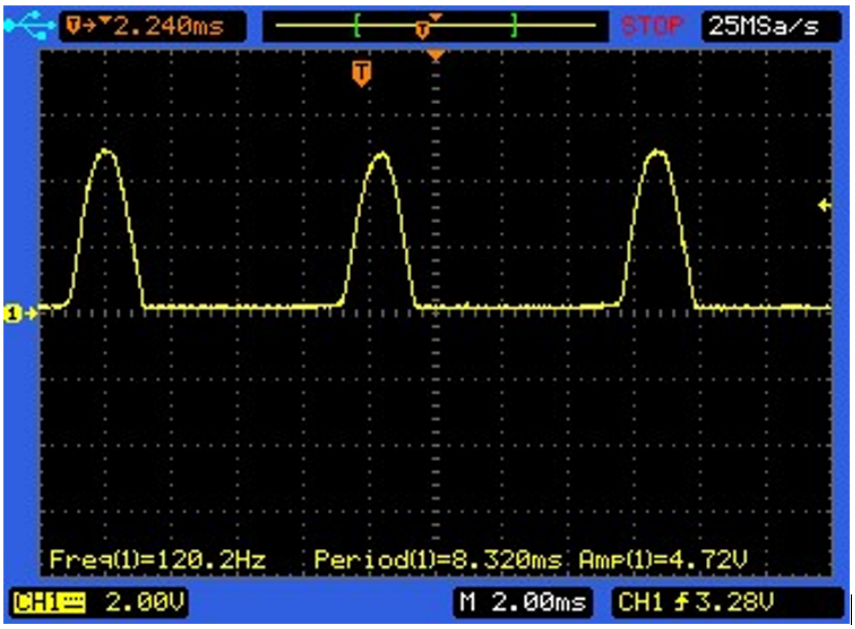


Figura 4.8: Prueba de verificación del circuito detector de cruce por cero.

El optoacoplador H11AA2X envía pulsos cada vez que la señal de voltaje alterno llega al valor de 0V AC, tanto la frecuencia como el periodo de los pulsos coinciden con la señal de voltaje alterno.

Al tener éxito con el circuito detector de cruce por cero se realiza la conexión de la salida al puerto digital 7 de la GrovePi+ mediante el cable universal de 4 pines grove con el fin de preparar la señal en el momento en que se ejecute el controlador PID desde la interfaz del sistema de monitoreo. Este circuito está directamente relacionado con el código de A.6 más específicamente con la instrucción condicional *while* encargada de realizar la sincronización de la señal de voltaje alterno que repercutirá con el bombillo cerámico emisor de calor.

4.3.2. Circuito de la etapa de potencia.

Se llevo a cabo la fabricación del circuito de control de potencia en PCB con los siguientes materiales y componentes electrónicos:

- 1 Placa de cobre 10x10 cm.
- 1 amplificador operacional TL084CN.
- TRIAC BT139.
- 2 borneras de 2 vias.
- 1 bornera de 3 vias.
- 1 optoacoplador MOC3021.

- 1 diodo emisor de luz.
- 3 resistencias de 100Ω.
- 2 resistencias de 10KΩ.

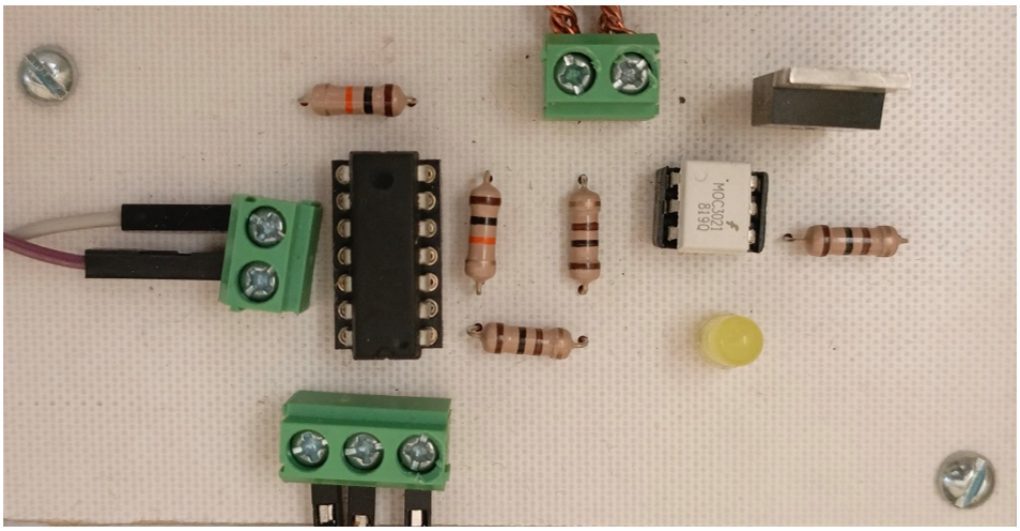


Figura 4.9: Elaboración del circuito de control de potencia.

Amplificación de la señal de control PWM.

En la Figura 4.10 se puede visualizar la forma en que la señal de control trabaja a una frecuencia de 119.0 Hz. Sin embargo, la tensión de la señal no es suficiente para utilizarse por completo al momento de trabajar en conjunto con el optoacoplador MOC3021.

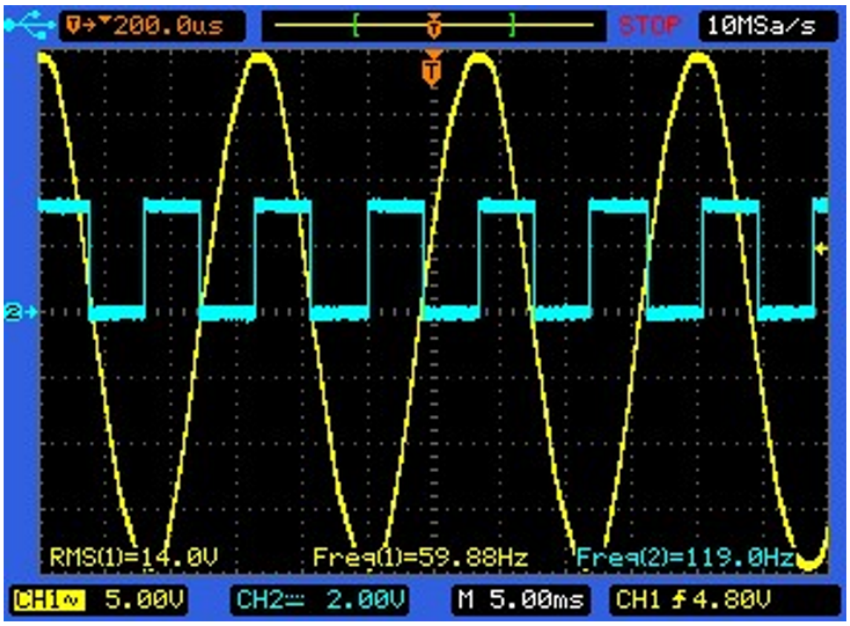
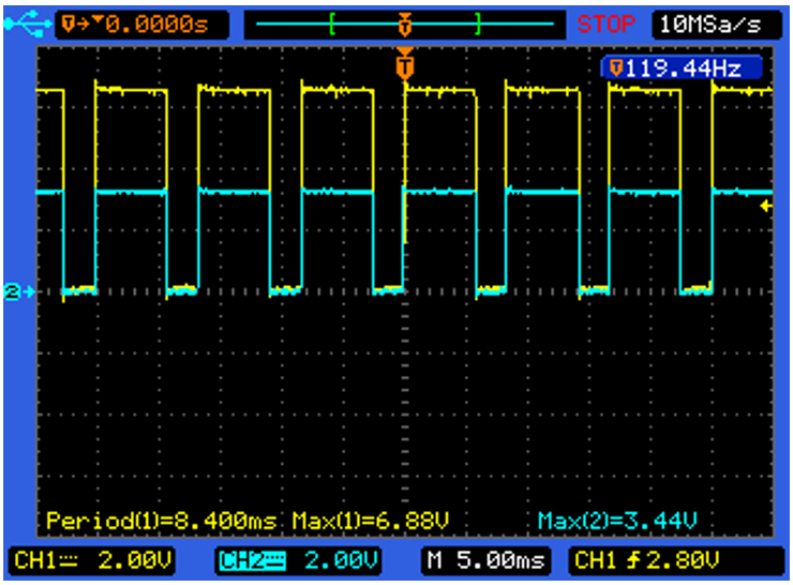
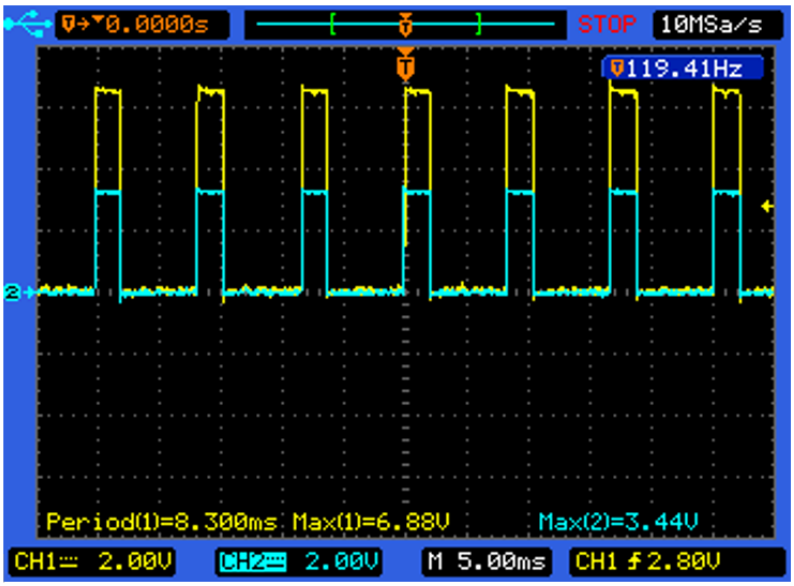


Figura 4.10: Pulso PWM acondicionada para trabajar con la señal de 120v AC.

Por esta razón la etapa de amplificación se lleva a cabo por medio del amplificador operacional TL084CN, al implementarse en configuración de amplificador no inversor el voltaje de salida encontrado fue de 6.88V. Es gracias a la Figura 4.11a y 4.11b que es posible observar el cambio en el ciclo de trabajo de la señal, primero con 80 % en flanco alto y 20 % en flanco bajo, se realiza el cambio después a 25 % en flanco alto y 75 % en flanco bajo. con una diferencia de 100mS en el periodo de la señal. La señal resultante de la amplificación posee bajo ruido y el voltaje no sufre variación alguna por lo que es adecuada para el optoacoplador MOC3021.



(a) Amplificación prueba 1.



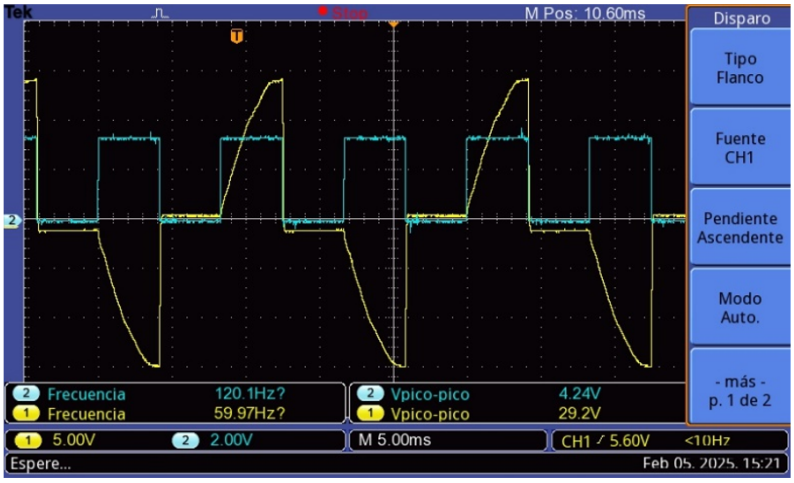
(b) Amplificación prueba 2.

Figura 4.11: Pruebas de amplificación del PWM proveniente del pin 18 de la Raspberry Pi.

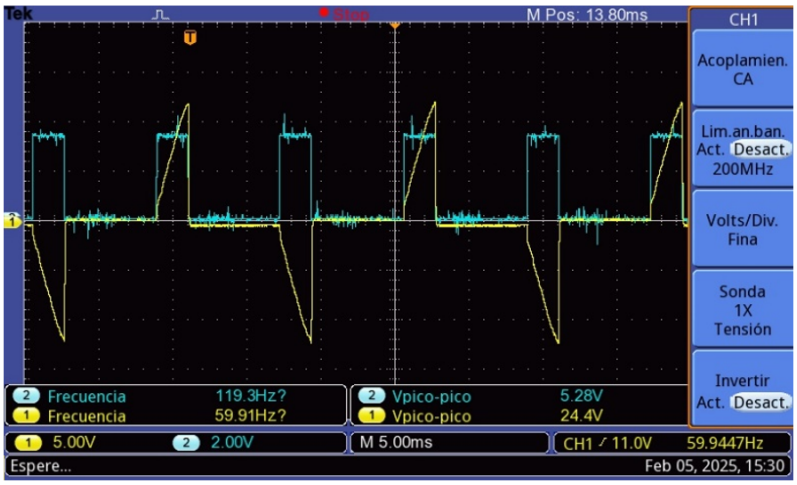
Control de la etapa de potencia del bombillo cerámico emisor de calor.

Una vez comprobado el funcionamiento del circuito detector de cruce por cero y la ampliación de la señal de control PWM, se realiza la validación del circuito de la etapa de potencia para observar en el osciloscopio el comportamiento real de la señal de corriente alterna actuante en el bombillo cerámico.

En la Figura 4.12a se puede observar la señal de control al 50 % del ciclo de trabajo, el comportamiento de la señal de corriente alterna se corta una vez que el PWM tiene flancos bajos y permitiendo su paso en flancos altos demostrando la sincronía que se obtuvo en las pruebas experimentales. Al variar el ciclo de trabajo (al 25 % en la Figura 4.12b ocurre lo mismo a pesar de la diferencia de 0.8Hz en la señal de control que hay entre una prueba y otra. Por otro lado se encontró menor diferencia (0.06Hz) en las señales de corriente alterno.



(a) 50 % de ciclo de trabajo.



(b) 25 % de ciclo de trabajo.

Figura 4.12: Pruebas de funcionamiento del circuito de la etapa de potencia.

4.4. Filtro de Kalman y PID de temperatura.

Dado que se realizaron cálculos para dos controladores de temperatura, en esta sección se presentan los resultados obtenidos de acuerdo a diferentes *setpoints* colocados desde la interfaz gráfica del controlador. En total se presentan 4 lotes de pruebas, el primer lote contiene 3 pruebas considerando solo un valor de referencia, el segundo lote 1 prueba con dos valores de referencia, un tercer lote de 1 prueba con 4 valores de referencia y un ultimo lote de 9 valores de referencia. Estos lotes son necesarios para verificar el comportamiento del PID simple y el PID+KF en distintas condiciones de operación (calentamiento y enfriamiento), la forma de evaluar su rendimiento es por medio de parámetros como el tiempo de retardo, el tiempo de subida, tiempo pico y porcentaje de sobreelongación (si es que aplica), el tiempo de asentamiento, la varianza en estado estable así como el error y temperatura promedio en estado estable. La forma de interpretar los valores de la varianza es que mientras mas pequeño menos dispersa es la temperatura captada muestra a muestra. En las Tablas 4.1, 4.2 y 4.3 se presenta un resumen del comportamiento de los controladores en el dominio del tiempo tales como el tiempo de retardo, de subida, si tuvo sobreelongación y el tiempo de asentamiento.

La interfaz gráfica del controlador fue fundamental para realizar estas pruebas, se siguieron los pasos siguientes para preparar la interfaz del controlador:

- Ejecutar el sistema de monitoreo desde el IDE Thonny IDE (código del anexo A.2).
- Seleccionar el sub menú del control PID.
- Digitar dentro de las cajas de texto; primero el valor de temperatura deseado, segundo, el valor de la ganancia proporcional, tercero, el valor de la ganancia integral y por ultimo el valor de la ganancia derivativa.
- Seleccionar el botón de inicio.
- Verificar el funcionamiento del controlador por medio de la actualización de las etiquetas situadas debajo de la leyenda "Parámetros".

En cada lote de pruebas los cálculos de varianza, el promedio de la temperatura en estado estable y el error se obtienen desde Matlab con ayuda de los siguientes comandos:

Promedios:

$$\text{promedio} = \text{mean}(tira\_datos)$$

Varianza:

$$\text{varianza} = \text{var}(tira\_datos)$$

Es importante contar con los archivos en excel para poder cargarlos en Matlab y posteriormente obtener las métricas. La variable *tira_ datos* contiene en una columna los datos a los que se les va a aplicar el promedio o la varianza.

Primer lote de pruebas.

Se realizan 3 pruebas en las que el controlador deberá llegar a distintas temperaturas, la temperatura inicial es de 17.79° C y el tiempo total que toma la prueba es de 2.5 hrs. por valor de referencia.

Para la primera prueba se decide asignar un valor de referencia de 22.0° C, en donde gracias a la Figura 4.13 se puede observar el comportamiento que tuvo utilizando ambos controladores, en conjunto con la Tabla 4.1 se encontró una mejor respuesta por parte del controlador PID apoyado por el filtro de Kalman en cuanto a precisión en la temperatura y en el nivel de ruido en estado estable, a pesar de que el tiempo de estabilización es mayor a comparación del PID simple. En ninguno de los casos se encuentran sobreelongaciones. Se destaca que, para esta primera prueba se encontró una diferencia de 0.138° C entre ambos controladores, siendo la mayor diferencia de las tres pruebas.

Valor de referencia No. 1		
Parámetro	**Valor**	
Temperatura inicial	20.11° C	
Setpoint	22.0° C	
Métricas en el dominio de tiempo	**PID**	**PID + KF**
Valor promedio alcanzado en estabilidad	22.167° C	22.014° C
Error promedio absoluto en estado estable	0.168° C	0.030° C
Tiempo de retardo	25.5 min.	19.83 min.
Tiempo de subida	57.5 min.	1.72 hrs.
Tiempo pico	-	-
Sobreelongación	-	-
Tiempo de asentamiento	1.42 hrs.	1.75 hrs.
Varianza en estado estable	0.00263	0.00112

Tabla 4.1: Resumen de resultados con el primer valor de referencia (Lote 1).

Una segunda prueba se realiza con una temperatura deseada de 24.0° C, Para esta prueba se encontró una mejor respuesta por parte del controlador PID+KF en cuanto a precisión de la temperatura en estado estable, al igual que logra un menor tiempo de asentamiento con un promedio de error menor al PID simple. Tampoco se encuentran sobreelongaciones.

Valor de referencia No. 2		
Parámetro	**Valor**	
Setpoint	24.0° C	
Métricas en el dominio de tiempo	**PID**	**PID + KF**
Valor promedio alcanzado en estabilidad	24.05° C	23.968° C
Error promedio absoluto en estado estable	0.058° C	0.045° C
Tiempo de retardo	22.66 min.	25.33 min.
Tiempo de subida	1.91 hrs.	2.12 hrs.
Tiempo pico	-	-
Sobreelongación	-	-
Tiempo de asentamiento	1.91 hrs.	1.81 hrs.
Varianza en estado estable	0.00138	0.00151

Tabla 4.2: Resumen de resultados con el segundo valor de referencia (Lote 1).

La tercera prueba se lleva a cabo con un *setpoint* de 25.0° C en donde el PID+KF tiene mejor tiempo de respuesta con una notable diferencia con el PID simple de 28.8 minutos para llegar al estado estable. El PID simple tiene mejor precisión con apenas 0.062° C de diferencia. En ninguno de los casos se encuentran sobreelongaciones.

Valor de referencia No. 3		
Parámetro	**Valor**	
Setpoint	25.0° C	
Métricas en el dominio de tiempo	**PID**	**PID + KF**
Valor promedio alcanzado en estabilidad	25.033° C	25.095° C
Error promedio absoluto en estado estable	0.061° C	0.098° C
Tiempo de retardo	22.0 min.	26.16 min.
Tiempo de subida	2.16 hrs.	1.65 hrs.
Tiempo pico	-	-
Sobreelongación	-	-
Tiempo de asentamiento	2.16 hrs.	1.68 hrs.
Varianza en estado estable	0.00119	0.00227

Tabla 4.3: Resumen de resultados con el tercer valor de referencia (Lote 1).

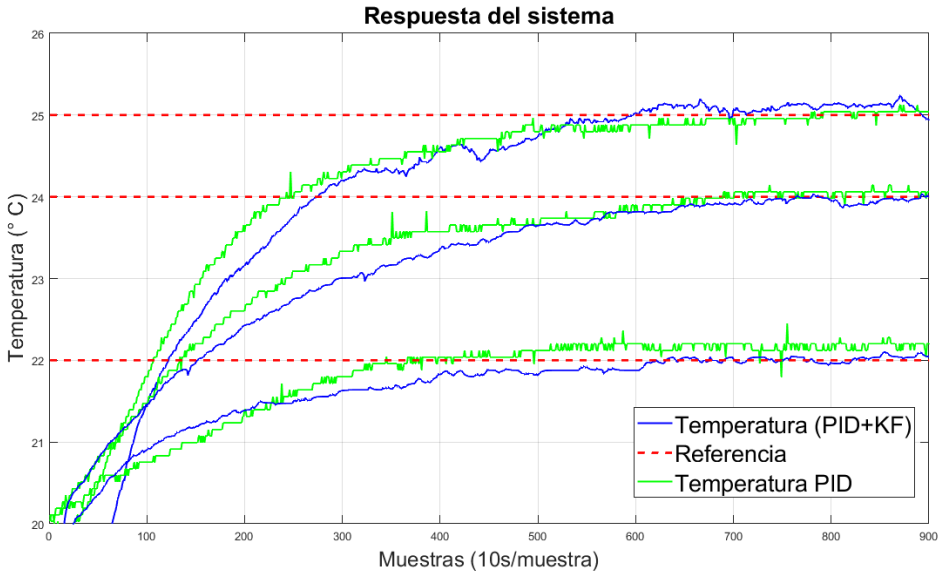


Figura 4.13: Controladores PID de temperatura con 1 valor de referencia (Lote 1).

Terminadas las 3 pruebas del primer lote, cuando se obliga al controlador PID a llegar a un solo valor de referencia, el filtro de Kalman resulta mejorar en mayor medida el tiempo de asentamiento, sin embargo, afecta en la estabilización dado que en 2 de 3 pruebas se encontró mayor varianza en las mediciones en estado estable. El promedio de temperatura en estado estable en ambos controladores es muy preciso, pese a que en el gráfico 4.13 el PID simple presenta picos en la temperatura monitoreada. El error promedio y la varianza de la temperatura son cercanas a cero por lo que para un invernadero de dimensiones pequeñas el control de la temperatura es preciso cuando no se somete a cambios bruscos en los valores de referencia.

Segundo lote de pruebas.

Para este segundo lote de pruebas se somete a los controladores PID a dos valores de referencia, en la Figura 4.14 se presenta el comportamiento de la temperatura según el controlador utilizado. El tiempo que toma esta prueba es de 8.13 horas.

Gráficamente el control PID+KF supera notoriamente al PID simple mas que nada en el tiempo de respuesta. En la Tabla 4.4 se puede observan incluso que el PID+KF presenta mejor precisión en la temperatura promedio alcanzada en estabilidad, una diferencia de 0.78 hrs. (46.8 minutos) de tiempo de asentamiento menos que el PID simple y apenas una diferencia de 0.003° C de error promedio sobre el valor deseado.

Valor de referencia No. 1		
Parámetro	**Valor**	
Temperatura inicial	20.9° C	
Setpoint	22.5° C	
Métricas en el dominio de tiempo	**PID**	**PID + KF**
Valor promedio alcanzado en estabilidad	22.443° C	22.546° C
Error promedio absoluto en estado estable	0.056° C	0.059° C
Tiempo de retardo	1.17 hrs.	22.83 min.
Tiempo de subida	2.66 hrs.	1.96 hrs.
Tiempo pico	-	-
Sobreelongación	-	-
Tiempo de asentamiento	2.66 hrs.	1.88 hrs.
Varianza en estado estable	0.00144	0.00313

Tabla 4.4: Resumen de resultados con el primer valor de referencia (Lote 2).

Una vez logrado el primer valor de referencia se realiza el cambio a una temperatura de 25.5° C (sin necesidad de detener la ejecución de la interfaz del controlador). En esta prueba el PID+KF supera en todos los aspectos al PID simple desde la precisión en la temperatura alcanzada en estabilidad, error promedio y menos tiempo de asentamiento (28.8 minutos menos). Tampoco se encuentran sobreelongaciones en ninguno de los dos casos.

Valor de referencia No. 2		
Parámetro	**Valor**	
Setpoint	25.5° C	
Métricas en el dominio de tiempo	**PID**	**PID + KF**
Valor promedio alcanzado en estabilidad	25.476° C	25.482° C
Error promedio absoluto en estado estable	0.045° C	0.030° C
Tiempo de retardo	1.86 hrs.	51.83 min.
Tiempo de subida	4.09 hrs.	3.48 hrs.
Tiempo pico	-	-
Sobreelongación	-	-
Tiempo de asentamiento	4.09 hrs.	3.48 hrs.
Varianza en estado estable	0.00178	0.00107

Tabla 4.5: Resumen de resultados con el segundo valor de referencia (Lote 2).

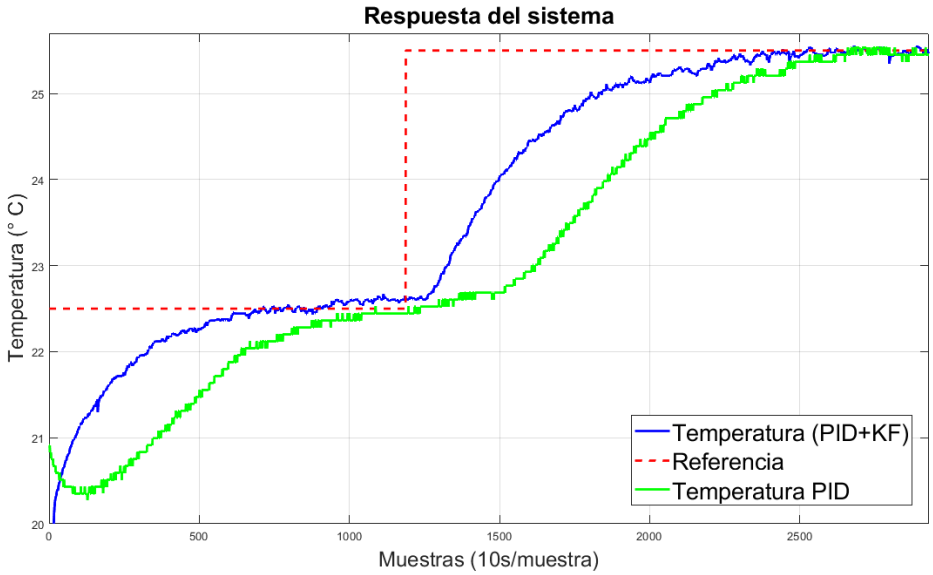


Figura 4.14: Controladores PID de temperatura con 2 valores de referencia (Lote 2).

Tercer lote de pruebas.

Se realiza una tercera prueba que consta de 4 valores de referencia en un tiempo de 34.47 horas (ver Figura 4.16).

A diferencia del lote 1 y lote 2 en donde se presento el calentamiento interno del invernadero, para el lote 3 se presentan métricas cuando se requiere bajar la temperatura. Los *setpoints* número 2 y 4 contenidos en las Tablas 4.7 y 4.9 corresponden al enfriamiento.

Primero se lleva la temperatura a un valor de 24.0° C, Gracias a la Tabla 4.6 se puede observar como el PID+KF tiene mejor precisión de la temperatura en estado estable, menor error promedio (0.03° C menos), pero un tiempo de estabilización mayor al PID simple (tarda 4.4 horas mas en estabilizarse).

Valor de referencia No. 1		
Parámetro	**Valor**	
Temperatura inicial	23.08° C	
Setpoint	24.0° C	
Métricas en el dominio de tiempo	**PID**	**PID + KF**
Valor promedio alcanzado en estabilidad	23.952° C	24.013° C
Error promedio absoluto en estado estable	0.062° C	0.032° C
Tiempo de retardo	49.83 min.	1.79 hrs.
Tiempo de subida	1.55 hrs.	5.95 hrs.
Tiempo pico	-	-
Sobreelongación	-	-
Tiempo de asentamiento	1.55 hrs.	5.95 hrs.
Varianza en estado estable	0.00489	0.00135

Tabla 4.6: Resumen de resultados con el primer valor de referencia (Lote 3).

Se lleva a cabo el cambio de la temperatura, en este caso baja de 24.0° C a 22.0° C. En la Tabla 4.7 se muestra como el PID simple tiene mejor precisión de la temperatura en estado estable y menor error promedio a comparación del PID+KF, sin embargo, el PID simple si presenta sobreelongaciones que alentan el tiempo de asentamiento provocando que tarde 2.44 hrs. mas que el PID+KF. En la Figura 4.15 se puede observar la sobreelongación del PID simple, y como tarda mas en converger al estado estable, caso contrario con lo que pasa en el PID+KF que inmediatamente converge en el primer instante que logra llegar al valor de referencia.

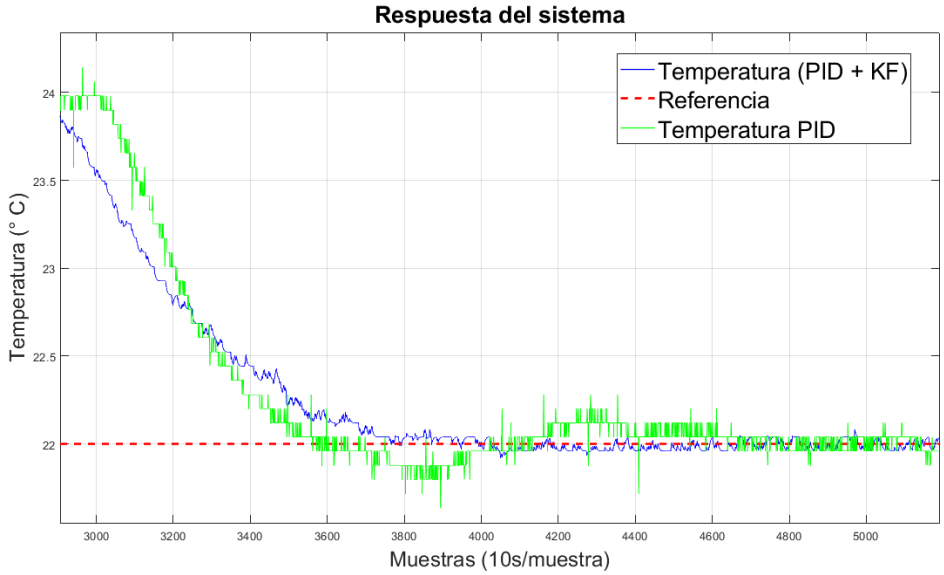


Figura 4.15: Temperatura en el valor de referencia número 2 (Lote 3).

Valor de referencia No. 2		
Parámetro	**Valor**	
Setpoint	22.0° C	
Métricas en el dominio de tiempo	**PID**	**PID + KF**
Valor promedio alcanzado en estabilidad	22.013° C	21.953° C
Error promedio absoluto en estado estable	0.013° C	0.057° C
Tiempo de retardo	1.02 hrs.	51.5 min.
Tiempo de subida	2.00 hrs.	2.63 hrs.
Tiempo pico	2.78 hrs.	-
Sobreelongación	-0.90 %	-
Tiempo de asentamiento	5.075 hrs.	2.63 hrs.
Varianza en estado estable	0.00328	0.00209

Tabla 4.7: Resumen de resultados con el segundo valor de referencia (Lote 3).

El tercer valor de referencia vuelve a ser de calentamiento, pasar de 22.0° C a 25.5° C. La Tabla 4.8 presenta como el controlador PID+KF vuelve a tener mejor rendimiento en cuanto a menor error promedio y un tiempo de asentamiento menor al PID simple dado que este ultimo si vuelve a presentar sobreelongación.

Valor de referencia No. 3		
Parámetro	**Valor**	
Setpoint	25.5° C	
Métricas en el dominio de tiempo	**PID**	**PID + KF**
Valor promedio alcanzado en estabilidad	25.421° C	25.412° C
Error promedio absoluto en estado estable	0.104° C	0.088° C
Tiempo de retardo	1.03 hrs.	43.66 min.
Tiempo de subida	2.95 hrs.	4.41 hrs.
Tiempo pico	3.95 hrs.	-
Sobreelongación	0.78 %	-
Tiempo de asentamiento	5.24 hrs.	4.41 hrs.
Varianza en estado estable	0.0105	0.00171

Tabla 4.8: Resumen de resultados con el tercer valor de referencia (Lote 3).

El cuarto valor de referencia vuelve a ser de enfriamiento, esta vez se baja la temperatura de 25.5° C a 24.2° C. En la Tabla 4.9 es importante recalcar que el PID simple presenta un tiempo de asentamiento menor al PID+KF pero no al valor de

referencia deseado (0.28° C por encima). El PID+KF si lo hace en un tiempo de 6.52 hrs. así como también se logra observar un error promedio menor.

Valor de referencia No. 4		
Parámetro	**Valor**	
Setpoint	24.2° C	
Métricas en el dominio de tiempo	**PID**	**PID + KF**
Valor promedio alcanzado en estabilidad	24.488° C	24.211° C
Error promedio absoluto en estado estable	0.288° C	0.024° C
Tiempo de retardo	3.55 hrs.	4.55 hrs.
Tiempo de subida	No llega al 100 %	5.96 hrs.
Tiempo pico	-	-
Sobreelongación	-	-
Tiempo de asentamiento	4.55 hrs.	6.52 hrs.
Varianza en estado estable	0.00586	0.00224

Tabla 4.9: Resumen de resultados con el cuarto valor de referencia (Lote 3).

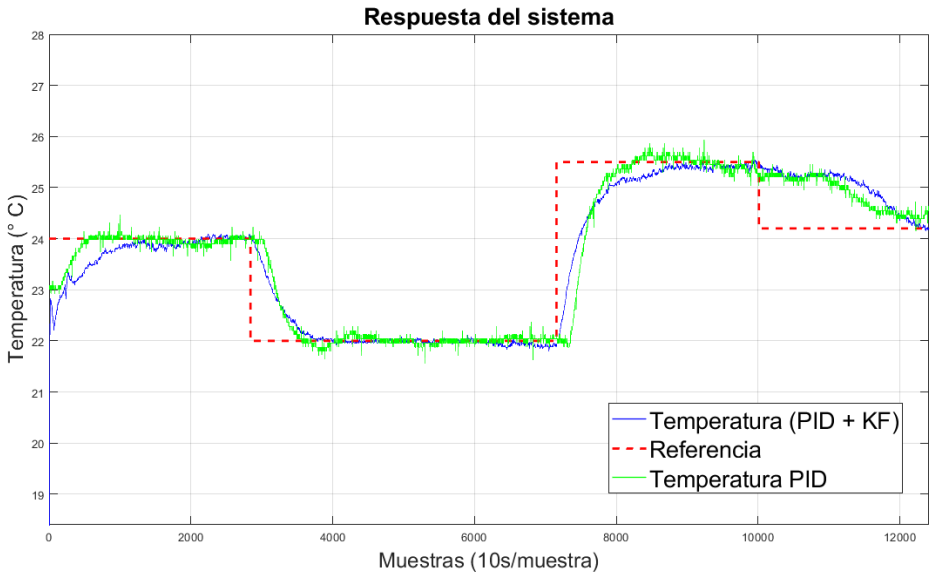


Figura 4.16: Controladores PID de temperatura con 4 valores de referencia (Lote 3).

Cuarto lote de pruebas.

Los valores de referencia sometidos en ambos controladores se presenta en la Figura 4.20, en total se muestra el cambio en la temperatura del invernadero con 9 valore de referencia en un tiempo de 77.7 horas.

El primer valor de temperatura es de 20.5° C con una temperatura inicial de 16.41° C. En la Tabla 4.10 se puede observar como el PID+KF se acerca mucho al valor deseado, la sobreelongación que presenta es mas pequeña que el PID simple (Figura 4.17) que, cabe destacar que no logra estabilizarse mostrando así oscilaciones prolongadas.

Valor de referencia No. 1		
Parámetro	**Valor**	
Temperatura inicial	16.41° C	
Setpoint	20.5° C	
Métricas en el dominio de tiempo	**PID**	**PID + KF**
Valor promedio alcanzado en estabilidad	21.169° C	20.493° C
Error promedio absoluto en estado estable	0.992° C	0.087° C
Tiempo de retardo	9.83 min.	2.16 min.
Tiempo de subida	46.66 min.	13.5 min.
Tiempo pico	1.2 hrs.	24.0 min.
Sobreelongación	11.85 %	3.21 %
Tiempo de asentamiento	No se estabiliza	32.5 min.
Varianza en estado estable	1.164	0.011

Tabla 4.10: Resumen de resultados con el primer valor de referencia (Lote 4).

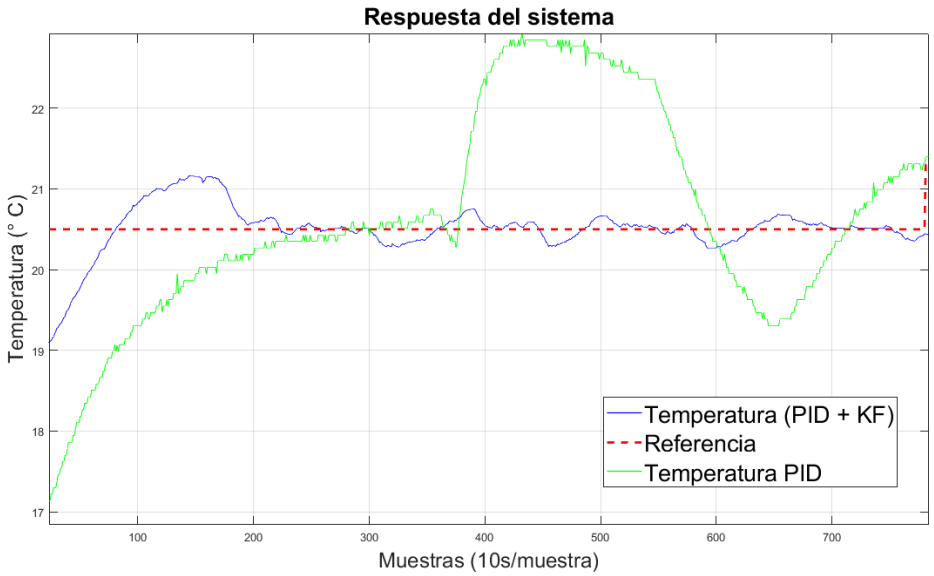


Figura 4.17: Temperatura del valor de referencia 1 (Lote 4)

Se lleva a cabo el segundo valor de referencia, el cual es pasar de 20.5° C a

21.3° C. Para esta prueba el PID simple tampoco logra estabilizarse además de presentar oscilaciones prolongadas, en la Tabla 4.11 se puede observar como para el PID simple se tiene datos como el tiempo de subida y el tiempo de retardo los cuales aparentan responder mejor que el PID+KF pero en el gráfico 4.18 se puede ver como unicamente se registra la primera oscilación después del cambio de temperatura.Por tanto el PID+KF realmente tiene mejor desempeño para este *setpoint*.

Valor de referencia No. 2		
Parámetro	**Valor**	
Setpoint	21.3° C	
Métricas en el dominio de tiempo	**PID**	**PID + KF**
Valor promedio alcanzado en estabilidad	21.289° C	21.310° C
Error promedio absoluto en estado estable	0.160° C	0.017° C
Tiempo de retardo	4.10 min.*	24.83 min.
Tiempo de subida	14.16 min.*	1.38 hrs.
Tiempo pico	Picos prolongados.	1.70 hrs.
Sobreelongación	2.34 %	2.72 %
Tiempo de asentamiento	No se estabiliza.	5.09 hrs.
Varianza en estado estable	0.0283	0.000782

Tabla 4.11: Resumen de resultados con el segundo valor de referencia (Lote 4).

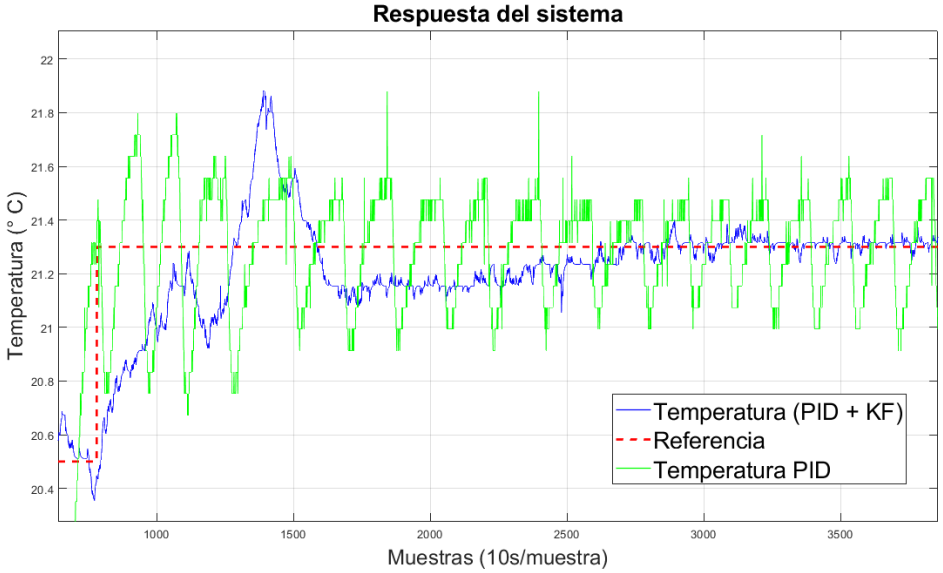


Figura 4.18: Temperatura del valor de referencia número 2 (Lote 4)

Para el tercer valor de referencia se aumaneta la temperatura de 21.3° C a 21.8° C. En el cual el PID simple converge mas rápido aunque el ruido provoca poca

estabilidad. El PID+KF tarda mas en converger pero llega a ser mas preciso. No se encuentran sobreelongaciones significativas en ninguno de los dos controladores.

Valor de referencia No. 3		
Parámetro	**Valor**	
Setpoint	21.8° C	
Métricas en el dominio de tiempo	**PID**	**PID + KF**
Valor promedio alcanzado en estabilidad	21.797° C	21.778° C
Error promedio absoluto en estado estable	0.051° C	0.040° C
Tiempo de retardo	4.5 min.*	39.66 min.
Tiempo de subida	15.33 min.*	2.00 hrs.
Tiempo pico	-	-
Sobreelongación	-	-
Tiempo de asentamiento	6.95 hrs.	1.98 hrs.
Varianza en estado estable	0.00299	0.00236

Tabla 4.12: Resumen de resultados con el tercer valor de referencia (Lote 4).

En el valor de referencia 4 el PID simple tampoco logra estabilizarse completamente, mientras que el el PID+KF sobrepasa el valor deseado por 0.043° C. En el control PID simple se vuelven a presentar sobreelongaciones tal como se muestra en la Figura 4.19.

Valor de referencia No. 4		
Parámetro	**Valor**	
Setpoint	21.5° C	
Métricas en el dominio de tiempo	**PID**	**PID + KF**
Valor promedio alcanzado en estabilidad	21.493° C	21.543° C
Error promedio absoluto en estado estable	0.162° C	0.047° C
Tiempo de retardo	4.83 min.*	40.83 min.
Tiempo de subida	5.5 min.*	2.01 hrs.
Tiempo pico	9.16 min.	-
Sobreelongación	-2 %	-
Tiempo de asentamiento	No se estabiliza.	2.96 hrs.
Varianza en estado estable	0.034	0.00096

Tabla 4.13: Resumen de resultados con el cuarto valor de referencia (Lote 4).

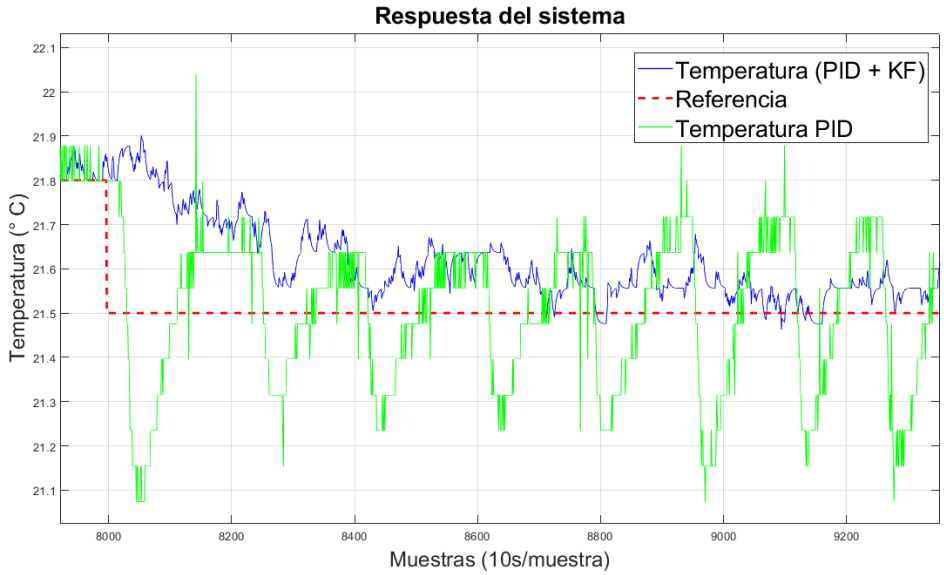


Figura 4.19: Temperatura del valor de referencia número 4 (Lote 4).

El siguiente valor de referencia es subir la temperatura de 21.5° C a 22.7° C. La Tabla 4.14 muestra como para esta prueba el PID simple logra un mejor rendimiento, desde la precisión de la temperatura alcanzada, el error promedio e incluso el tiempo de asentamiento.

Valor de referencia No. 5		
Parámetro	**Valor**	
Setpoint	22.7° C	
Métricas en el dominio de tiempo	**PID**	**PID + KF**
Valor promedio alcanzado en estabilidad	22.714° C	22.671° C
Error promedio absoluto en estado estable	0.036° C	0.171° C
Tiempo de retardo	24.0 min.	42.33 min.
Tiempo de subida	1.23 hrs.	3.73 hrs.
Tiempo pico	-	-
Sobreelongación	-	-
Tiempo de asentamiento	3.72 hrs.	3.73 hrs.
Varianza en estado estable	0.00288	0.00135

Tabla 4.14: Resumen de resultados con el quinto valor de referencia (Lote 4).

Ahora se sube la temperatura de 22.7° C a 23.4° C. en la Tabla 4.15 el PID simple vuelve a tener mejor rendimiento sobre el PID+KF, en donde el filtro logra mitigar el error promedio con una diferencia de 0.007° C.

Valor de referencia No. 6		
Parámetro	**Valor**	
Setpoint	23.4° C	
Métricas en el dominio de tiempo	**PID**	**PID + KF**
Valor promedio alcanzado en estabilidad	23.408° C	23.432° C
Error promedio absoluto en estado estable	0.057° C	0.050° C
Tiempo de retardo	25 min.	40 min.
Tiempo de subida	53.33 min.	2.13 hrs.
Tiempo pico	-	-
Sobreelongación	-	-
Tiempo de asentamiento	53.33 min.	2.13 hrs.
Varianza en estado estable	0.00467	0.00427

Tabla 4.15: Resumen de resultados con el sexto valor de referencia (Lote 4).

En el valor de referencia número 7 el PID aumenta la temperatura a los 24° C, se puede visualizar en la Tabla 4.16 como el mejor controlador es el PID simple.

Valor de referencia No. 7		
Parámetro	**Valor**	
Setpoint	24.0° C	
Métricas en el dominio de tiempo	**PID**	**PID + KF**
Valor promedio alcanzado en estabilidad	24.031° C	24.015° C
Error promedio absoluto en estado estable	0.043° C	0.065° C
Tiempo de retardo	1.30 hrs.	1.13 hrs.
Tiempo de subida	2.50 hrs.	3.0 hrs.
Tiempo pico	2.86 hrs.	-
Sobreelongación	0.95 %	-
Tiempo de asentamiento	4.19 hrs.	4.38 hrs.
Varianza en estado estable	0.00364	0.00587

Tabla 4.16: Resumen de resultados con el séptimo valor de referencia (Lote 4).

En los últimos 3 cambios de referencia el PID simple demostró tener un mejor rendimiento pese a las sobreelongaciones que presenta, mientras mas se decide aumentar la temperatura al PID+KF le toma mas tiempo converger al valor de referencia. En la prueba 8 se baja la temperatura de 24.0° C a 23.6° C. Nuevamente

el PID simple sobresale en el tiempo de asentamiento, error promedio y precisión en la temperatura.

Valor de referencia No. 8		
Parámetro	**Valor**	
Setpoint	23.6° C	
Métricas en el dominio de tiempo	**PID**	**PID + KF**
Valor promedio alcanzado en estabilidad	23.569° C	23.547° C
Error promedio absoluto en estado estable	0.046° C	0.065° C
Tiempo de retardo	25 min.	1.30 hrs.
Tiempo de subida	48.33 min	3.38 hrs.
Tiempo pico	-	-
Sobreelongación	-	-
Tiempo de asentamiento	48.33 min.	3.38 hrs.
Varianza en estado estable	0.00400	0.00404

Tabla 4.17: Resumen de resultados con el octavo valor de referencia (Lote 4).

El ultimo *setpoint* se presenta en la Tabla 4.18, en donde el PID+KF logra un menor tiempo de converger aunque por precisión y error promedio el PID simple resulta ser mejor.

Valor de referencia No. 9		
Parámetro	**Valor**	
Setpoint	24.5° C	
Métricas en el dominio de tiempo	**PID**	**PID + KF**
Valor promedio alcanzado en estabilidad	24.460° C	24.441° C
Error promedio absoluto en estado estable	0.052° C	0.060° C
Tiempo de retardo	50 min.	1.33 hrs.
Tiempo de subida	2.0 hrs.	7.19 hrs.
Tiempo pico	3.91 hrs.	-
Sobreelongación	0.85 %	-
Tiempo de asentamiento	8.41 hrs.	7.19 hrs.
Varianza en estado estable	0.00222	0.00167

Tabla 4.18: Resumen de resultados con el noveno valor de referencia (Lote 4).

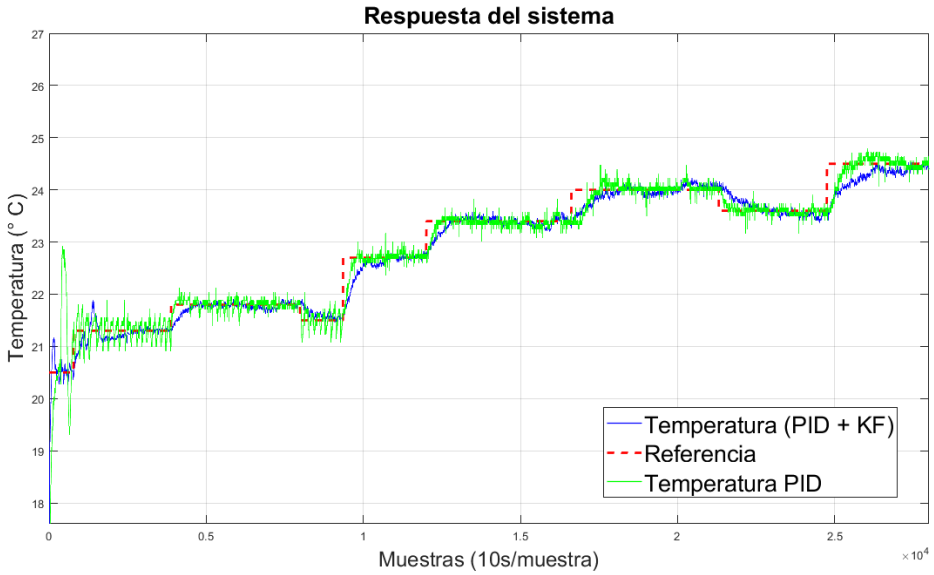


Figura 4.20: Controladores PID de temperatura con 9 valores de referencia (Lote 4).

Adicionalmente se presentan gráficas de convergencia del error, dado que también es un indicador para evaluar el rendimiento del filtro sobre el controlador.

La primera se puede visualizar en la Figura 4.21, en la que se puede encontrar 3 líneas correspondientes al error en cada una de las pruebas del PID de los primeros 3 resultados de resultados presentados en esta sección. Conforme pasa el tiempo el error disminuye hasta converger a un valor muy cercano a cero, mientras más cercano a cero más estabilidad logra el controlador.

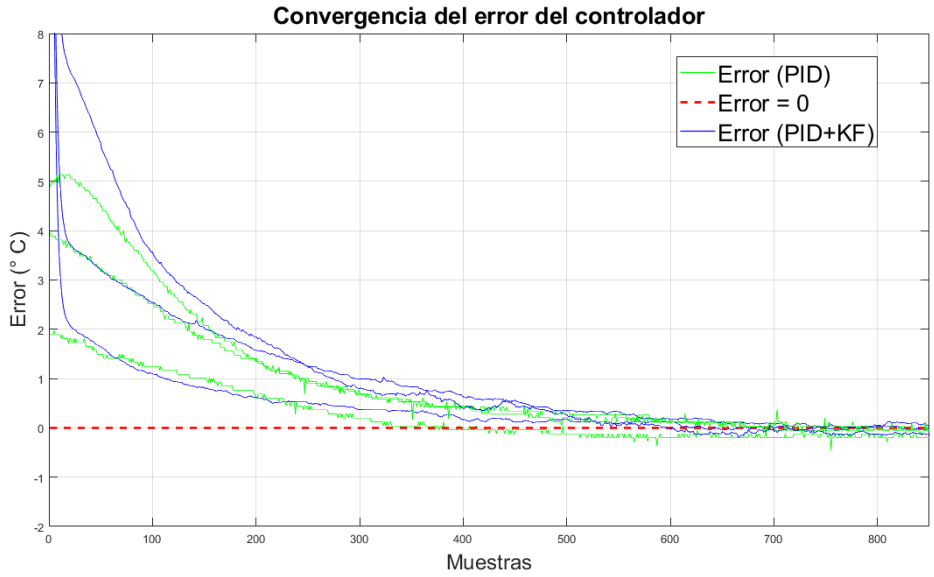


Figura 4.21: Convergencia del error del control PID (Lote 1).

En la Figura 4.22 se puede visualizar la convergencia del error de las pruebas

realizadas y presentadas en la Figura 4.14, el efecto del filtro auxilia al controlador PID para mejorar el rendimiento, el error converge en menos tiempo.

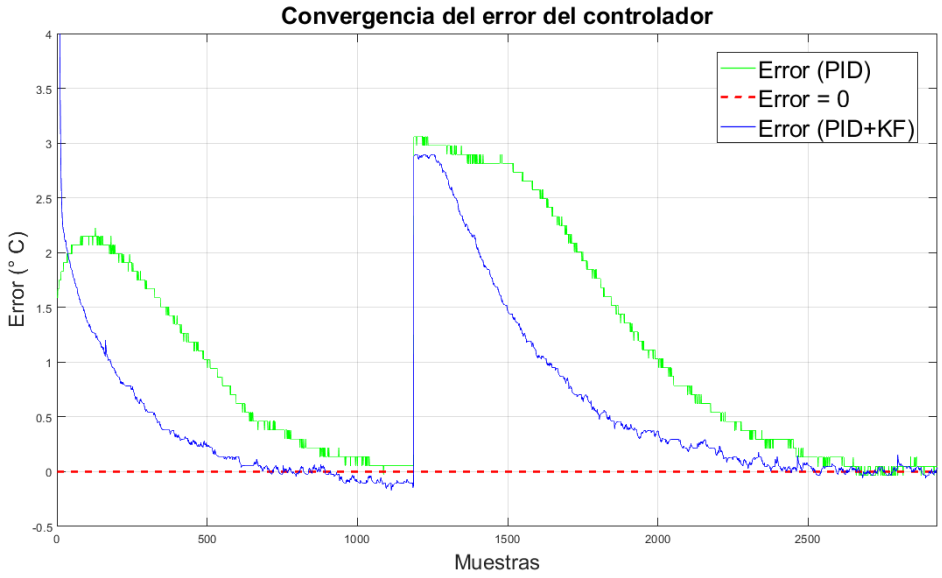


Figura 4.22: Convergencia del error del controlador (Lote 2).

Gráficamente las respuestas presentadas en el caso del uso del filtro son favorables en comparación del controlador sin el uso de él.

El impacto del filtro en la señal de control.

De acuerdo con los últimos lotes de pruebas mostrados anteriormente, se pudieron observar en su mayoría como a comparación del control PID simple, en el PID+KF las sobreelongaciones eran menos comunes. Esto directamente se puede llegar a observar con la evolución de la señal de control. Para ello se presenta la señal del controlador PID (acotado entre 0-1000) y del PWM (0 % a 100 %) del lote de pruebas número 2.

Tal y como se presenta en la Figura 4.23 se puede observar 4 gráficas en las que se presenta la señal del controlador y el PWM que se genera para el actuador térmico (gráficas del lado izquierdo). Por otro lado en las gráficas del lado derecho corresponden a las señales de control cuando interviene el filtro de Kalman en las lecturas del sensor de temperatura antes de entrar a procesarse en el modulo del controlador PID.

Del gráfico, se puede observar como el PID simple arroja picos en la señal de control que influye directamente en el PWM generado por la Raspberry Pi. Para ello basta con visualizar los valores de PWM entre 0 y 4 en el eje x, en donde, el PWM se alterna constantemente. Por otro lado en el PID+KF en ese mismo lapso de tiempo la señal de control es mas "suave"lo que genera un PWM con menos alternancia.

El que se este generando un PWM mas constante afecta directamente al actuador, ya que físicamente se evita estrés mecánico lo cual alarga la vida útil del actuador

térmico. Además, una señal PWM suave representa una mayor eficiencia energética reduciendo las perdidas por conmutación en transistores y circuitos de potencia. Otra de las ventajas que ofrece el filtro es la reducción de armónicos, de manera que se pueda evitar posibles interferencias en otros dispositivos electrónicos cercanos.

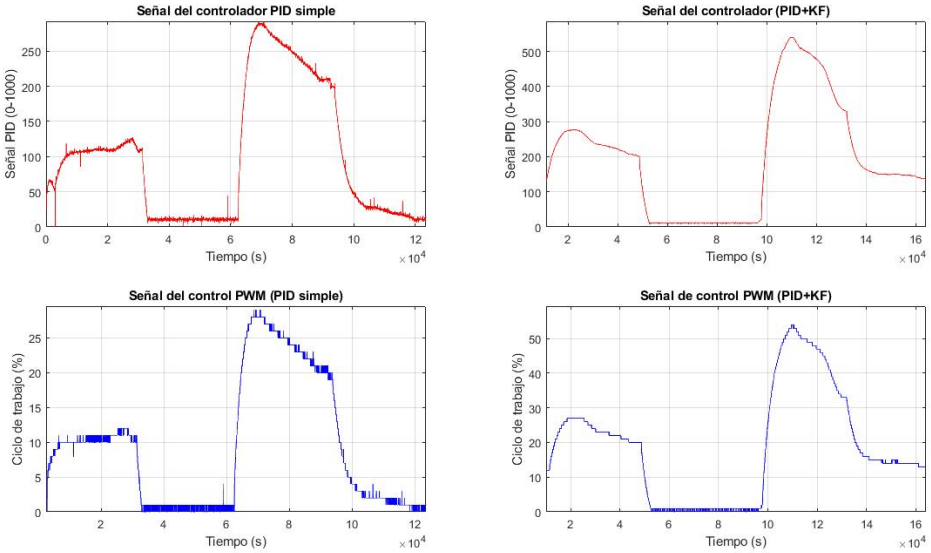


Figura 4.23: Señales de control (Lote 3)

Es así que el filtro actúa sobre el controlador de manera que representa mas beneficios a pesar de demorar mas tiempo en asentarse al valor de referencia.

Por último, se realiza un comparativo con 4 criterios principales, mayor precisión, error promedio menor, mejor tiempo de asentamiento y menor varianza en estado estable de todas las pruebas realizadas en los 4 lotes. En la Tabla 4.19 se puede observar que el filtro mejora el rendimiento del controlador PID en cuanto a mitigar el error promedio en estado estable en 11 de 18 pruebas y lograr un tiempo mas rápido de asentamiento en 12 de 18 pruebas sobre el PID simple.

Criterio	**PID simple**	**PID+KF**
Mayor precisión	9	9
Error promedio menor	7	11
Mejor tiempo de asentamiento	6	12
Menor varianza en estado estable	6	12

Tabla 4.19: Comparativo final

Conclusiones

Se realizó la implementación de un filtro de Kalman en un controlador PID de temperatura a partir de datos experimentales que representa el comportamiento de la temperatura en un tiempo de 2 horas tomando muestras cada 10 segundos mediante el uso del sensor analógico de temperatura "*Grove temperature sensor v1.2*".

Se obtuvieron 2 controladores PID, cada uno con ganancias P, I y D distintas. El primer controlador se obtuvo de acuerdo con los parámetros τ y L de la curva de reacción con la etapa de filtrado mientras que para el segundo se omite el filtro de Kalman y se calculan las ganancias con la señal ruidosa. Se encontró una mayor diferencia entre las ganancias proporcionales (4.0688) a comparación de la ganancia integral (0.008196) y derivativa (0.0020485) de cada uno.

Se llevaron a cabo 18 pruebas, organizadas en 4 lotes, comparando el rendimiento PID simple con el PID+KF. Los resultados indican que el PID+KF mejoro la precisión de la temperatura en estado estable en el 50 % de las pruebas, disminuyo el error promedio en estado estable en 61.11 % y optimizo el tiempo de asentamiento en 66.66 % de los casos. Estos hallazgos demuestran que el filtro de Kalman aporta beneficios significativos al control PID.

Según los resultados obtenidos, la varianza del sistema es menor al emplear el filtro de Kalman, con un rango de 0.000782 a 0.011. En cambio, cuando el PID opera sin filtrado la varianza se amplia alcanzando valores entre 0.022 y 1.164. Estos datos evidencian que la implementación del filtro en el control PID mejora la estabilidad térmica del invernadero.

El filtro de Kalman además de eliminar ruido del sensor de temperatura *grove temperature sensor v1.2* tuvo un efecto positivo en el control PID en 3 aspectos; disminuyo el error promedio en estado estable, mejoro el tiempo de asentamiento y logro disminuir la varianza en los estados de temperatura.

Se logró el desarrollo e implementación de un sistema de monitoreo basado en interfaces gráficas de usuario (GUI) para la supervisión en tiempo real de temperatura, luz y humedad del suelo, cumpliendo con el objetivo planteado. Este sistema fue implementado en una Raspberry Pi 3 B+ en conjunto con un módulo GrovePi+, permitiendo la adquisición de datos de sensores y su representación visual. El uso de archivos .xlsx permitió gestionar de manera organizada los registros obtenidos, facilitando su análisis mediante software externo. La integración del controlador PID

dentro de la interfaz gráfica proporcionó un ajuste eficiente de los parámetros de control, permitiendo mejorar la estabilidad térmica del sistema.

Entre los desafíos encontrados, se observó que en las gráficas 2D un tiempo de actualización bajo altera la ejecución de otras interfaces, otro de los desafíos se presento al momento de manejar el tiempo de los eventos de las interfaces gráficas, problemas que fueron resueltos a través del análisis en los tiempos de actualización de cada interfaz. En futuras mejoras se podría incluir la optimización en la gestión de memoria y procesamiento de datos. También se recomienda la incorporación de alertas automáticas basadas en umbrales de temperatura, luz y humedad, así como la opción de exportar gráficos en formatos adecuados para reportes técnicos. En cuanto al rendimiento, al tener las interfaces abiertas y en ejecución continua el estado del procesador no rebasa el 10 % de uso, por lo que el software diseñado es optimo.

En conclusión, el sistema desarrollado cumplió con éxito su propósito de proporcionar una herramienta de monitoreo intuitiva y funcional, integrando la supervisión de variables ambientales y el control PID dentro de una plataforma eficiente basada en Raspberry Pi y GrovePi+.

Se diseñó, simuló y fabricó un circuito de control de potencia para un bombillo cerámico, utilizando una combinación de modulación por ancho de pulso (PWM) y un Triac. Para ello, se empleó un amplificador operacional TL084CN, un optoacoplador MOC3021 y un Triac BT139, con una fuente de 12V (S-60-12) para la etapa de control. En la fase experimental, el circuito logró regular la potencia del bombillo cerámico, permitiendo un control preciso de la temperatura. Se observaron diferencias menores entre la simulación y la implementación real, principalmente en los tiempos de conmutación del Triac, posiblemente debido a variaciones en los componentes y a la naturaleza de la conmutación en corriente alterna.

El método experimental de Ziegler-Nichols arrojo resultados alentadores. Al aplicar el filtro a la curva de reacción, logro mitigar las componentes ruidosas suavizando la señal sin presentar desfase. La ejecución del filtro no representa un obstáculo al momento de ejecutar la interfaz gráfica del controlador por lo cual permitió el uso del software libre de interrupciones.

De acuerdo a las pruebas realizadas en cada uno de los lotes, se encontró que el invernadero es idóneo para mantener temperaturas dentro del siguiente rango: de 16.4° C hasta los 26.0°C con una precisión de ±0.035° C cuando se utiliza con el filtro de Kalman y ±0.087° C cuando no se usa.

Bibliografía

[1] NMSC, "Mexico Industrial Process Automation Market Analysis - 2030," 2 2024. [Online]. Available: https://www.nextmsc.com/report/mexico-industrial-process-automation-market

[2] R. C. Dorf and R. H. Bishop, *Modern control systems.* Prentice Hall, 1 2011.

[3] K. Ogata, *Sistemas de control en tiempo discreto 2 ed.* Pearson Educación, 1 1996.

[4] B. Kuo, *Sistemas de control automático.* Prentice Hall Hispanoamericana, 1996. [Online]. Available: https://books.google.com.mx/books?id=GyWr6cT8SEsC

[5] M. Melda Ulusoy, "An optimal state estimator understanding kalman filters, part 3," 2017. [Online]. Available: https://la.mathworks.com/videos/understanding-kalman-filters-part-3-optimal-state-estimator--1490710645421.html?s_tid=srchtitle_videos_main_7_kalman%20filter

[6] A. Becker, *Kilman filter.* Alex Becker, 1 2023.

[7] D. Molloy, *Exploring Raspberry Pi.* John Wiley & Sons, 6 2016.

[8] R. P. Pinout, "GPIO 18 (PCM Clock)," s.f. [Online]. Available: https://pinout.xyz/pinout/5v_power

[9] HYUNDAI, "Micro SDHC Hyundai 16 GB," s. f. [Online]. Available: https://hyundaitechnology.com.mx/shop/sdc16gu1-new-micro-sdhc-hyundai-16-gb-139#attr=

[10] S. Studio, "Grove - Universal 4 pin 20cm Unbuckled Cable (5 PCs Pack)," s.f. [Online]. Available: https://www.seeedstudio.com/Grove-Universal-4-Pin-20cm-Unbuckled-Cable-5-PCs-Pack-p-749.html

[11] W. S. Studio, "Grove Ecosystem Introduction | Seeed Studio Wiki," 3 2023. [Online]. Available: https://wiki.seeedstudio.com/Grove_System/

[12] S. Studio, "Grove - temperature sensor," s.f. [Online]. Available: https://www.seeedstudio.com/Grove-Temperature-Sensor.html

[13] S. Studio., "Grove - Soil moisture sensor," s.f. [Online]. Available: https://www.seeedstudio.com/Grove-Moisture-Sensor.html

[14] S. Studio, "Grove - Light Sensor v1.2 - LS06-S Phototransistor Compatible with Arduino," s.f. [Online]. Available: https://www.seeedstudio.com/Grove-Light-Sensor-v1-2-LS06-S-phototransistor.html

[15] A. H. Donate, *Electrónica aplicada.* MARCOMBO, 1 2013.

[16] Onsemi, *6-Pin DIP Random-Phase Triac Driver Output Optocoupler (250/400 V Peak)*, Onsemi, 2023. [Online]. Available: https://www.onsemi.com/download/data-sheet/pdf/moc3023m-d.pdf

[17] ISOCOM Components, *A.C. INPUT PHOTOTRANSISTOR OPTICALLY COUPLED ISOLATORS*, ISOCOM Components, 2008. [Online]. Available: http://isocom.com/wp-content/uploads/2017/08/H11AA1X_2X_3X_4X.pdf

[18] R. M. Marston, *Optoelectronics Circuits Manual.* Butterworth-Heinemann, 1 1996.

[19] WcEn Semiconductor, *BT139-800*, WcEn Semiconductor, 2018. [Online]. Available: https://www.mouser.com/datasheet/2/848/BT139_800-3104514.pdf

[20] NextHermal, "Elstein ceramic infrared," 4 2025. [Online]. Available: https://nexthermal.com/elstein-ceramic-ir/

[21] T. J. Maloney, *Electrónica industrial moderna*, 5th ed. Pearson Educación, 1 2006.

[22] T. Instruments, *TL08xx FET-Input Operational Amplifiers*, Texas Instruments, 2024. [Online]. Available: https://www.ti.com/product/TL084/part-details/TL084CN

[23] M. M. WELL, *60W Single Output Switching Power Supply*, MW MEAN WELL, 2012. [Online]. Available: https://www.mouser.mx/datasheet/2/260/mean_well_mwec-s-a0000136639-1-1737958.pdf

[24] Thonny, "Thonny, Python IDE for beginners," s.f., https://thonny.org/.

[25] D. Industries, "Install the Image on The Raspberry Pi," 1 2021. [Online]. Available: https://www.dexterindustries.com/howto/install-raspbian-for-robots-image-on-an-sd-card/

[26] K. Ogata, *Ingeniería de control moderna 5 ed.* PRENTICE HALL, 1 2010.

[27] A. Kumar, V. Singh, S. Kumar, S. P. Jaiswal, and V. S. Bhadoria, "Iot enabled system to monitor and control greenhouse," *Materials Today: Proceedings*, vol. 49, pp. 3137–3141, 2022, national Conference on Functional Materials: Emerging Technologies and Applications in Materials Science, https://doi.org/10.1016/j.matpr.2020.11.040. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214785320386363

[28] M. Valipour, K. M. Toffolo, and L. A. Ricardez-Sandoval, "State estimation and sensor location for Entrained-Flow Gasification Systems using Kalman Filter," *Control Engineering Practice*, vol. 108, p. 104702, 12 2020. [Online]. Available: https://doi.org/10.1016/j.conengprac.2020.104702

[29] S. Wakitani, H. Nakanishi, Y. Ashida, and T. Yamamoto, "Study on a Kalman Filter based PID Controller," *IFAC-PapersOnLine*, vol. 51, no. 4, pp. 422–425, 1 2018. [Online]. Available: https://doi.org/10.1016/j.ifacol.2018.06.131

[30] M. Contini, G. Orrù, A. A. Sini, M. Sole, and D. D. Giusto, "The ayo! project for air quality monitoring," in *2016 IEEE International Symposium on Systems Engineering (ISSE)*, 2016, pp. 1–4, 10.1109/SysEng.2016.7753171.

[31] C. Balasubramaniyan and D. Manivannan, "IoT Enabled Air Quality Monitoring System (AQMS) using Raspberry Pi," *Indian Journal of Science and Technology*, vol. 9, no. 39, 10 2016. [Online]. Available: https://doi.org/10.17485/ijst/2016/v9i39/90414

[32] A. Math, L. Ali, and U. Pruthviraj, "Development of smart drip irrigation system using iot," in *2018 IEEE Distributed Computing, VLSI, Electrical Circuits and Robotics (DISCOVER)*, 2018, pp. 126–130, 10.1109/DISCOVER.2018.8674080.

[33] T. J. E. Andrés, "Sistema de control de software e interfaz de usuario para control de microclimas de cultivo con gateway iot," 2016. [Online]. Available: http://repositorio.unab.cl/xmlui/handle/ria/3726

[34] A. Nagarajan, P. Nagarajan, M. Neeruganti, V. Teja, and R. Suresh, "Intelligent greenhouse monitoring and controlling by using python on raspberry pi," *Efflatounia*, pp. 2667–2673, 09 2021, iSSN: 1110-8703.

[35] J. Cañadas, J. A. Sánchez-Molina, F. Rodríguez, and I. M. del Águila, "Improving automatic climate control with decision support techniques to minimize disease effects in greenhouse tomatoes," *Information Processing in Agriculture*, vol. 4, no. 1, pp. 50–63, 2017, https://doi.org/10.1016/j.inpa.2016.12.002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214317316300737

[36] P. Cepeda, P. Ponce, A. Molina, and E. Lugo, "Towards sustainability of protected agriculture: Automatic control and structural technologies integration of an intelligent greenhouse," *IFAC Proceedings Volumes*, vol. 46, no. 7, pp. 366–371, 2013, 11th IFAC Workshop on Intelligent Manufacturing Systems, https://doi.org/10.3182/20130522-3-BR-4036.00085. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1474667015357025

[37] G. R. J. German, "Diseño de un prototipo de invernadero automatizado e implementación de estrategias de control PID y On-Off para el control de temperatura y humedad, supervisadas por medio de la plataforma ThingSpeak y almacenamiento de datos en MySQL," 3 2021. [Online]. Available: http://repositorio.uan.edu.co/handle/123456789/2291

[38] S. Romadhon and A. Multi, "Design and Development of Real-Time Monitoring & Controlling Infant Incubator with Tilt Stabilizer Using Raspberry Pi Remotely Controlled via PC and Smartphone to Reduce Tilt during Baby Transfer," *International Journal of Advanced Multidisciplinary*, vol. 2, no. 2, pp. 569–580, 9 2023. [Online]. Available: https://doi.org/10.38035/ijam.v2i2.310

[39] I. Firmansyah, W. A. Sukarto, B. Hermanto, J. M. Asta, and H. Nugraha, "Wi-fi based temperature monitoring system for thermal analysis with kalman filter implementation," in *2016 International Conference on Computer, Control, Informatics and its Applications (IC3INA)*, 2016, pp. 1–5, 10.1109/IC3INA.2016.7863013.

[40] Z. Gao, L. He, and X. Yue, "Design of pid controller for greenhouse temperature based on kalman," in *Proceedings of the 3rd International Conference on Intelligent Information Processing*, ser. ICIIP '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 1–4. [Online]. Available: https://doi.org/10.1145/3232116.3232117

[41] E. A. Abioye, M. S. Z. Abidin, M. S. A. Mahmud, S. Buyamin, O. D. Ijike, A. O. Otuoze, A. A. Afis, and O. M. Olajide, "A data-driven kalman filter-pid controller for fibrous capillary irrigation," *Smart Agricultural Technology*, vol. 3, p. 100085, 2023, https://doi.org/10.1016/j.atech.2022.100085. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2772375522000508

[42] V. M. Carlos, *Sistemas de control continuos y discretos*. Ediciones Paraninfo, S.A., 1 2012.

[43] W. Altmann, "8 - tuning of pid controllers in both open and closed loop control systems," in *Practical Process Control for Engineers and Technicians*, W. Altmann, D. Macdonald, and S. Mackay, Eds. Oxford: Newnes, 2005, pp. 112–130, https://doi.org/10.1016/B978-075066400-4/50008-2. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780750664004500082

[44] F. Miyara, "Filtros activos," *Universidad Nacional de Rosario Facultad de Ciencias Exactas, Ingeniería y Agrimensura Escuela de Ingeniería Electrónica*, 2004. [Online]. Available: https://www.fceia.unr.edu.ar/enica3/filtros-t.pdf

[45] G. L. Medel J. J., Pedro, and A. Flores Rueda, "Caracterización de filtros digitales en tiempo real para computadoras digitales," *Computación y Sistemas, 7(3), 190-209*, 2004. [Online]. Available: http://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S1405-55462004000100006&lng=es&tlng=es.

[46] G. Ellis, "Chapter 9 - filters in control systems," in *Control System Design Guide*, 4th ed., G. Ellis, Ed. Boston: Butterworth-Heinemann, 2012, pp. 165–183. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780123859204000096

[47] A. Zaknich, *Principles of Adaptive Filters and Self-learning Systems*. Springer Science & Business Media, 4 2005.

[48] L. C. Barrios and J. J. P. Álvarez, *Filtros para relés digitales de protección de sistemas eléctricos*. Editorial Universitaria (Cuba), 8 2020.

[49] D. Simon, *Optimal state estimation*. John Wiley & Sons, 6 2006.

[50] D. Stranneby and W. Walker, "6 - introduction to kalman filters," in *Digital Signal Processing and Applications (Second Edition)*, 2nd ed., D. Stranneby and W. Walker, Eds. Oxford: Newnes, 2004, pp. 159–177. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780750663441500069

[51] A. Pajankar, A. Kakkar, M. Poole, and R. Grimmett, *Raspberry Pi: Amazing Projects from Scratch*. Packt Publishing Ltd, 9 2016.

[52] F. Oliveira, D. G. Costa, and I. Silva, "On the development of flexible mobile multi-sensor units based on open-source hardware platforms and a reference framework," *HardwareX*, vol. 10, p. e00243, 2021, https://doi.org/10.1016/j.ohx.2021.e00243. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2468067221000730

[53] J. S. Wilson, *Sensor Technology Handbook*. Newnes, 1 2005.

[54] Ó. R. Jiménez, *Python a fondo*. Marcombo, 2 2021.

[55] N. Mohan, T. M. Undeland, and W. P. Robbins, *Power Electronics*, 3rd ed. John Wiley & Sons Inc., 2003.

[56] M. Granados, *Electrónica y Microcontroladores PIC: Aprende electrónica básica y microcontroladores PIC en un lenguaje sencillo*. Misael Granados, 2020. [Online]. Available: https://books.google.com.mx/books?id=N0q5EAAAQBAJ

[57] V. Badarinath, "How do Ceramic Infrared Heaters Actually Work?" 7 2022. [Online]. Available: https://nexthermal.com/how-do-ceramic-infrared-heaters-actually-work/

[58] A. Annamaa, "Introducing thonny, a python ide for learning programming," in *Proceedings of the 15th Koli Calling Conference on Computing Education Research*, ser. Koli Calling '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 117–121. [Online]. Available: https://doi.org/10.1145/2828959.2828969

Anexos

Anexo A: Scripts del sistema de monitoreo.

A.1. Adquisición de datos provenientes de los sensores.

Nombre del programa: valor_sensores.py

```
import grovepi
import math
import time
potentiometer=0
def lectura_hummodulo(moisture_sensor):
    sensor_value2 = grovepi.analogRead(moisture_sensor)
    return (sensor_value2)

def lectura_lummodulo(light_sensor):
    sensor_value1 = grovepi.analogRead(light_sensor)
    return (sensor_value1)

def lectura_temp(temp):
    sensor_value = grovepi.temp(potentiometer,'1.2')
    return(sensor_value)
```

A.2. Programa principal.

Nombre del programa: interfaz_principal.py

```
import grovepi
import pid
import valor_sensores
import graficos
import impresion_datos
import grabado_datos
from tkinter import *
import tkinter as tk
from tkinter import ttk

i=0
```

```
12 temp=0
13 light_sensor=1
14 moisture_sensor=2
15 sensor_temp=4
16
17
18
19
20 salida_cero=7
21 controluz=8
22 grovepi.pinMode(controluz,"OUTPUT")
23 grovepi.pinMode(motor_agua,"OUTPUT")
24 grovepi.pinMode(temp,"INPUT")
25 grovepi.pinMode(light_sensor,"INPUT")
26 grovepi.pinMode(moisture_sensor,"INPUT")
27 grovepi.pinMode(salidacero,"INPUT")
28
29
30 root=tk.Tk()
31 root.title('Principal')#Esta linea tambien funciona con root.
      wm_title('SENSORES')
32 app=impresion_datos.App(root)
33 root.geometry("300x240")
34 label=Label(root, text="MENU")
35 label.pack(pady=10)
36 label.place(x=130, y=110)
37 sum_err=0
38
39 img=tk.PhotoImage(file="temp.png")
40 loadimg=tk.Label(root, image=img)
41 loadimg.pack()
42 loadimg.place(x=57,y=22)
43
44 img2=tk.PhotoImage(file="humedad.png")
45 loadimg2=tk.Label(root, image=img2)
46 loadimg2.pack()
47 loadimg2.place(x=149,y=22)
48
49 img3=tk.PhotoImage(file="luz.png")
50 loadimg3=tk.Label(root, image=img3)
51 loadimg3.pack()
52 loadimg3.place(x=228,y=22)
53
54
55 btn1=tk.Button(root,
56                text="1.PID",
57                command=lambda: pid.APP2(root))
58 btn1.pack()
59 btn1.place(x=10,y=135)
60
61 btn3=Button(root,
62             text="2.Graficos",
63             command=lambda: graficos.graf())
64 btn3.pack()
65 btn3.place(x=75, y=135)
66
```

```
btn2=tk.Button(root,
               text="3.Grabar_Datos",
               command=lambda: grabado_datos.start(root))
btn2.pack()
btn2.place(x=170, y=135)

btn4=tk.Button(root,
               text="Cerrar",
               bg="red",
               foreground="white",
               command = root.destroy)
btn4.pack()
btn4.place(x=120, y=170)


root.mainloop()
```

A.3. Impresión de lecturas de los sensores, iconos y rutina de control para el encendido y apagado de luces.

Nombre del programa: impresión_datos.py

```
import grovepi
import valor_sensores
from tkinter import *
import tkinter as tk
from tkinter import ttk
import time
from datetime import datetime

temp=0
light_sensor=1
moisture_sensor=2
controluz=8

hiluz= "09:23"
hfluz= "17:23"

class App:
    def __init__ (self,master):
        self.master=master
        frame=Frame(master)
        frame.pack()
        label_main=Label(frame,
                         text='SENSORES',
                         font=("Helvetica",15))

        label_main.grid(row=0)

        labeln1=Label(frame,
```

```
                text='')

        labeln1.grid(row=1,
                     column=1)

        labeln2=Label(frame,
                      text='')

        labeln2.grid(row=2,
                     column=1)


        label=Label(frame,
                    text='Temperatura',
                    font=("Helvetica", 10))

        label.grid(row=3,
                   column=0)

        self.reading_label=Label(frame,
                                 text='12.34',
                                 font=("Arial", 13))

        self.reading_label.grid(row=4,
                                column=0)

        label2=Label(frame,
                     text='Humedad',
                     font=("Helvetica",10))

        label2.grid(row=3,
                    column=1)

        self.reading_label2=Label(frame,
                                  text='12.34',
                                  font=("Arial", 13))

        self.reading_label2.grid(row=4,
                                 column=1)

        labeln=Label(frame,
                     text='')
        labeln.grid(row=1,
                    column=3)

        label3=Label(frame,
                     text='Luz',
                     font=("Helvetica",10))

        label3.grid(row=3,
                    column=4)

        self.reading_label1=tk.Label(frame,
                                     text='12.34',
                                     font=("Arial", 13))

```

```
85          self.reading_label1.grid(row=4,
86                                   column=4)
87          self.update_reading()
88
89      def update_reading(self):
90          val1=valor_sensores.lectura_lummodulo(light_sensor)
91          reading_str1="{:.2f}".format(val1)
92          self.reading_label1.configure(text=reading_str1)
93
94          val2=valor_sensores.lectura_hummodulo(moisture_sensor)
95          reading_str2="{:.2f}".format(val2)
96          self.reading_label2.configure(text=reading_str2)
97
98          val=valor_sensores.lectura_temp(temp)
99          reading_str="{:.2f}".format(val)
100         self.reading_label.configure(text=reading_str)
101
102         self.master.after(800,self.update_reading)
103         self.master.after(1000,self.luces_crecimiento)
104
105     def luces_crecimiento(self):
106         Hora_actual=datetime.now().strftime(%H:%M)
107         if hora_actual==hiluz:
108             grovepi.digitalWrite(controluz.1)
109         if hora_actual>=hiluz and hora_actual<hfluz:
110             grovepi.digitalWrite(controluz.1)
111         if hora_actual==hfluz:
112             grovepi.digitalWrite(controluz.0)
```

A.4. Interfaz de gráficos de las variables: temperatura, luz y humedad de suelo.

Nombre del programa: "gráficos.py"

```
1  import grovepi
2  import valor_sensores
3  import time
4  import collections
5  import matplotlib.pyplot as plt
6  import matplotlib.animation as animation
7  from matplotlib.lines import Line2D
8
9  temp=0
10 light_sensor=1
11 moisture_sensor=2
12
13
14 Samples = 100  #Muestras
15 sampleTime = 90000  #Tiempo de muestreo
16
17 class graf:
18     def __init__(self):
```

```
19          xmin = 0
20          xmax = Samples
21          ymin = [0, 0 , -50 ,0]
22          ymax = [985, 985 , 50 , 100]
23          lines = []
24          data = []
25
26          fig = plt.figure()# Crea una nueva figura
27          ax1 = fig.add_subplot(2, 2, 1,
28                                xlim=(xmin, xmax),
29                                ylim=(ymin[0] , 40))
30          ax1.title.set_text('Temperatura')
31          ax1.set_xlabel("Tiempo")
32          ax1.set_ylabel("Temperatura(C)")
33
34          ax2 = fig.add_subplot(2, 2, 2,
35                                xlim=(xmin, xmax),
36                                ylim=(0 , 1000))
37          ax2.title.set_text('Luz')
38          ax2.set_xlabel("Tiempo")
39          ax2.set_ylabel("Nivel_de_luz")
40
41          ax3 = fig.add_subplot(2, 2, 3,
42                                xlim=(xmin, xmax),
43                                ylim=(0 , 600))
44          ax3.title.set_text('Humedad')
45          ax3.set_xlabel("Tiempo")
46          ax3.set_ylabel("Humedad")
47
48          def CapturaDeValores(self,Samples,lines):
49
50              valorhum= grovepi.analogRead(moisture_sensor)
51              data[2].append(valorhum)
52              lines[2].set_data(range(Samples),data[2])
53              time.sleep(.2)
54
55              valorluz= grovepi.analogRead(light_sensor)
56              data[1].append(valorluz)
57              lines[1].set_data(range(Samples),data[1])
58              time.sleep(.2)
59
60              value  = valor_sensores.lectura_temp(temp)
61              data[0].append(value)
62              lines[0].set_data(range(Samples),data[0])
63              time.sleep(.2)
64
65          for i in range (0,3):
66              data.append(collections.deque([0] * Samples, maxlen=
                    Samples))
67              lines.append(Line2D([], [], color='red'))
68
69          ax1.add_line(lines[0])
70          ax2.add_line(lines[1])
71          ax3.add_line(lines[2])
72
```

```
73          anim = animation.FuncAnimation(fig,CapturaDeValores, fargs
               =(Samples,lines), interval=sampleTime)
74          plt.show()
```

A.5. Modulo del controlador PID digital.

Nombre del programa: “modulopid.py”

```
1  def controlador(setpoint, p, i, d, sf, z, erra):
2
3      now_err = setpoint - sf
4      z =z + now_err
5
6
7      now_val = p * (now_err) + i * z + d * (now_err - erra)
8      last_err = now_err
9
10     if (now_val<0):
11         now_val=0
12     if (now_val>1000):
13         now_val=1000
14
15     return now_val,z,now_err,last_err
```

A.6. Interfaz gráfica del controlador PID y etapa de filtrado.

Nombre del programa: “pid.py”

```
1  import grovepi
2  import modulopid
3  from tkinter import *
4  import tkinter as tk
5  from tkinter import ttk
6  import valor_sensores
7  import math
8  import RPi.GPIO as GPIO
9
10 '''VARIABLES FILTRO KALMAN'''
11
12 A=1
13 Q=0.01
14 R=1.0968
15 X_n=0
16 P_n=0
17
18 ''' FIN VARIABLES FILTRO KALMAN'''
19
```

```
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
pwm_pin=18
GPIO.setup(pwm_pin, GPIO.OUT)
freq_Hz=120.0

pwm=GPIO.PWM(pwm_pin,freq_Hz)
pwm.start(0)
sum_err=0
temp=0
salidacero=7
val=0
last_erri=0

val_pid=0
val_pwm=0
class APP2:
    def __init__(self,root):

        '''Desarrollo de la interfaz'''

        def cerrar_pid():
            nw.destroy()

        var1=tk.StringVar()
        var2=tk.StringVar()
        var3=tk.StringVar()
        var4=tk.StringVar()

        nw=tk.Toplevel(root)
        self.nw=nw
        nw.title("PID")
        nw.geometry("350x500+500+100")
        Label(nw,
              text="Sintonizacion",
              font=("Helvetica",15)).pack()

        entry=ttk.Entry(nw,
                        width=10)
        entry.insert(0,"")
        entry.place(x=100,
                    y=50)
        setpoint=ttk.Button(nw,
                            text="Setpoint",
                            command=lambda: entry.get())
        setpoint.place(x=210,
                       y=50)

        entry2=ttk.Entry(nw,
                         width=10)
        entry2.insert(0,"")
        entry2.place(x=100,
                     y=90)
        proporcional=ttk.Button(nw,
                                text="P",
                                command=lambda: entry2.get())
```

```
76          proporcional.place(x=210,
77                             y=90)
78
79          entry3=ttk.Entry(nw,
80                           width=10)
81          entry3.insert(0,"")
82          entry3.place(x=100,
83                       y=130)
84          integral=ttk.Button(nw,
85                              text="I",
86                              command=lambda: entry3.get())
87          integral.place(x=210,
88                         y=130)
89
90          entry4=ttk.Entry(nw,
91                           width=10)
92          entry4.insert(0,"")
93          entry4.place(x=100,
94                       y=170)
95          derivativo=ttk.Button(nw,
96                                text="D",
97                                command=lambda: entry4.get())
98          derivativo.place(x=210,
99                           y=170)
100
101         guardar=ttk.Button(nw,
102                            text="Iniciar",
103                            command=lambda: capturar())
104         guardar.place(x=210,
105                       y=210)
106
107         cerrar=ttk.Button(nw,
108                           text="Cerrar",
109                           command=lambda: cerrar_pid())
110         cerrar.place(x=100,
111                      y=210)
112
113         label2=tk.Label(nw,
114                         text="Pararametros",
115                         font=("Helvetica",15))
116         label2.place(x=120,
117                      y=250)
118
119         label1=tk.Label(nw,
120                         text= "Temperatura:")
121         label1.place(x=100,
122                      y=290)
123
124         label3=tk.Label(nw,
125                         text= "Setpoint:")
126         label3.place(x=100,
127                      y=320)
128
129         label4=tk.Label(nw,
130                         text= "Sum_err:")
131         label4.place(x=100,
```

```
                y=350)

        label5=tk.Label(nw,
                        text= "PWM:")
        label5.place(x=100,
                y=380)

        label6=tk.Label(nw,
                        text= "Error:")
        label6.place(x=100,
                y=410)

        self.readinglabelapp2=tk.Label(nw,
                                       text="0.0")
        self.readinglabelapp2.place(x=210,y=290)

        self.readinglabel2app2=tk.Label(nw,
                                        text="0.0")
        self.readinglabel2app2.place(x=210,y=320)

        self.readinglabel3app2=tk.Label(nw,
                                        text="0.0")
        self.readinglabel3app2.place(x=210,
                                     y=350)

        self.readinglabel4app2=tk.Label(nw,
                                        text="0.0")
        self.readinglabel4app2.place(x=210,
                                     y=380)

        self.readinglabel5app2=tk.Label(nw,
                                        text="0.0")
        self.readinglabel5app2.place(x=210,
                                     y=410)

        self.act_tempid()

        def capturar():
            global sum_err, last_erri, val_pid, val_pwm
            global val

            '''VARIABLES DEL FILTRO DE KALMAN'''

            global shat
            global X_n
            global P_n

            '''FIN VARIABLES DEL FILTRO KALMAN'''

            if (entry.get()!=""):
                var1=float(entry.get())
            var2=float(entry2.get())
            var3=float(entry3.get())
            var4=float(entry4.get())
            sensor_value_app2= valor_sensores.lectura_temp(temp)

```

```
188             '''FILTRO KALMAN'''
189
190             X_n = A * X_n
191             P_n = A**2 * P_n + Q
192             error = sensor_value_app2 -X_n
193             K = P_n / ( P_n + R )
194             X_n = X_n + K * error
195             P_n = ( 1 - K) * P_n
196
197             shat = X_n
198
199             nuevo_valor,sum_err,now_err,last_erri=modulopid.
                    controlador(var1,
200                                                                 var2,
201                                                                 var3,
202                                                                 var4,
203                                                                 shat,
204                                                                 sum_err
                                                                        )
205
206             j=int(nuevo_valor*100/1000)
207             hora_inte=time.strftime("%X")
208             capturapidpwm.almacen_pid_pwm(nuevo_valor,
209                                           j,
210                                           sensor_value_app2,
211                                           shat,
212                                           hora_inte)
213
214             acot1="{:.2f}".format(nuevo_valor)
215             acot2="{:.2f}".format(now_err)
216             acot3="{:.2f}".format(var1)
217             while val==1:
218                 val=grovepi.digitalRead(salidacero)
219
220             pwm.start(j)
221
222             if (var1!=""):
223
224                 self.readinglabel2app2.config(text=(acot3))
225
226                 self.readinglabel3app2.config(text=(acot1))
227                 self.readinglabel4app2.config(text=(j))
228
229                 self.readinglabel5app2.config(text=(acot2))
230             nw.after(800,capturar)
231
232
233     def act_tempid(self):
234         valp = valor_sensores.lectura_temp(temp)
235         acot0="{:.2f}".format(valp)
236         self.readinglabelapp2.config(text=(acot0))
237         self.nw.after(400,self.act_tempid)
```

A.7. Herramienta de grabado de datos de temperatura.

Nombre del programa: “grabado_datos.py”

```
import grovepi
import valor_sensores
import time
import xlsxwriter
import tkinter as tk
from tkinter import *
import capturapidpwm


'''VALORES DEL FILTRO DE KALMAN'''
A=1
Q=0.01
R=1.0968
X_n=0
P_n=1
'''                                '''
v=1
i=0
h=0
v_e=0
w=0

temp=0
grovepi.pinMode(temp,"INPUT")

cont_archivos=[]
cont_hojas=[]
stop=0

intgd_pid=[]
intgd_pwm=[]
intgd_temp=[]
intgd_fk=[]
intgd_tiempo=[]
tmwin=0
class start:

    def __init__(self,root):

        def guardarnombre():
            new= entry.get()
            archivo=xlsxwriter.Workbook(str(new))
            cont_archivos.append(archivo)
            hoja=archivo.add_worksheet()
            cont_hojas.append(hoja)

        def stopdata():
            global i
            global h
```

```
50              i=0
51              h=0
52              intgd_pid,intgd_pwm,intgd_temp,intgd_fk,intgd_tiempo=
                    capturapidpwm.vaciado()
53              tam1=len(intgd_pid)
54              tam2=len(intgd_pwm)
55              if(tam1<=tam2):
56                  tmwin=1
57              else:
58                  tmwin=0
59              y=cont_hojas[-1]
60              if(tmwin==1):
61                  for p in range (0,tam2):
62                      contpid=intgd_pid[p]
63                      contpwm=intgd_pwm[p]
64                      conttemp=intgd_temp[p]
65                      contfk=intgd_fk[p]
66                      contmarct=intgd_tiempo[p]
67
68                      y.write(p,3,contpid)
69                      y.write(p,4,contpwm)
70                      y.write(p,5,conttemp)
71                      y.write(p,6,contfk)
72                      y.write(p,7,contmarct)
73              else:
74                  for p in range (0,tam1):
75                      contpid=intgd_pid[p]
76                      contpwm=intgd_pwm[p]
77                      conttemp=intgd_temp[p]
78                      contfk=intgd_fk[p]
79                      contmarct=intgd_tiempo[p]
80
81                      y.write(p,3,contpid)
82                      y.write(p,4,contpwm)
83                      y.write(p,5,conttemp)
84                      y.write(p,6,contfk)
85                      y.write(p,7,contmarct)
86
87              r=cont_archivos[-1]
88              r.close()
89              nueva_ventana(self)
90              nw12.destroy()
91
92
93          def nueva_ventana(self):
94              nw13=tk.Toplevel(root)
95              self.nw13=nw13
96              nw13.title("RECORD")
97              nw13.geometry("235x100")
98              Label(nw13,
99                    text="DATOS␣GUARDADOS",
100                   font=("Helvetica",10)).place(x=50, y=15)
101
102             cerrar=ttk.Button(nw13,
103                               text="Aceptar",
104                               command=nw13.destroy)
```

```
105             cerrar.place(x=75, y=50)
106
107         nw12=tk.Toplevel(root)
108         self.nw12=nw12
109         nw12.title("RECORD")
110         nw12.geometry("350x300+500+100")
111         Label(nw12,
112               text="TEMPERATURA",
113               font=("Helvetica",15)).place(x=100,y=25)
114
115         entry=ttk.Entry(nw12,width=25, justify=tk.CENTER)
116         entry.insert(0,".xlsx")
117         entry.place(x=75,y=85)
118         Nombre=ttk.Button(nw12,
119                           text="Crear␣Archivo",
120                           command=lambda: guardarnombre())
121         Nombre.place(x=135, y=115)
122
123         label1=tk.Label(nw12,text= "TEMPERATURA:")
124         label1.place(x=100,y=200)
125
126         label2=tk.Label(nw12,text= "MUESTRAS:")
127         label2.place(x=100,y=230)
128
129         capt=ttk.Button(nw12,text="Iniciar",command=lambda:
                startdata())
130         capt.place(x=75, y=155)
131
132         guardar=ttk.Button(nw12,text="Finalizar",command=lambda:
                stopdata())
133         guardar.place(x=185, y=155)
134
135         self.readinglabelapp2=tk.Label(nw12,text="0.0")
136         self.readinglabelapp2.place(x=210,y=200)
137
138         self.readinglabelapp3=tk.Label(nw12,text="0.0")
139         self.readinglabelapp3.place(x=210,y=230)
140
141
142         def startdata():
143             global i
144             global h
145             global v_e
146             global shat
147             global X_n
148             global P_n
149             global w
150
151             val2=valor_sensores.lectura_temp(temp)
152
153             X_n = A * X_n
154             P_n = A**2 * P_n + Q
155
156             error = val2-X_n
157             K = P_n / ( P_n + R )
158             X_n = X_n + K * error
```

```
159             P_n = ( 1 - K ) * P_n
160             shat = X_n
161 
162             hora=time.strftime("%X")
163 
164             reading_str="{:.3f}".format(val2)
165             t=float(reading_str)
166 
167             y=cont_hojas[-1]
168             y.write(w,2,shat)
169             y.write(i,1,t)
170             y.write(h,0,hora)
171 
172             h=h+1
173             i=i+1
174             w=w+1
175 
176             self.readinglabelapp2.config(text=(reading_str))
177             self.readinglabelapp3.config(text=(i))
178             nw12.after(10000,startdata)
```

Anexo B: Script que plotea la curva de reacción de la temperatura interna del invernadero.

B.1. Programa de la curva de reacción

Nombre del programa: "curva_reaccion.py"

```
1  import time
2  import xlsxwriter
3  import numpy as np
4  import matplotlib.pyplot as plt
5  from openpyxl import Workbook
6  from openpyxl import load_workbook
7  plt.style.use('ggplot')
8 
9  archivo=xlsxwriter.Workbook("Filtrokalman5.xlsx")
10 hoja=archivo.add_worksheet()
11 cont_hojas=[]
12 cont_hojas.append(hoja)
13 h=0
14 
15 N=674
16 A=1
17 R=1.0968
18 Q=0.01
19 X_n=0
20 P_n=1
21 
22 shat=np.zeros(N)
23 sr=np.zeros(N)
24 
```

```
value=[]
i=0
j=1
libro=load_workbook("Mier_20_09_23.xlsx")
hoja=libro.active

x=0
vector=[]
recta_tangente=[]

for i in range (0, 150):
    x=i
    vector.append(x)
    y7=0.110532*x+20.568739
    recta_tangente.append(y7)


for i in range (0,674):
    n=i
    value.append(float(hoja.cell(row=j ,
                                 column= 2).value))
    j=j+1
    sr[n]=value[n]

datv=sr
v=np.var(datv)

for n in np.arange(N):

    X_n = A * X_n
    P_n = A**2 * P_n + Q
    error = sr[n] - X_n
    K = P_n /( P_n + R )
    X_n = X_n + K * error
    P_n = ( 1 - K ) * P_n
    spred = X_n
    shat[n] = spred
    y = cont_hojas[-1]
    y.write(h,2,spred)
    h = h + 1

archivo.close()
print(v)
plt.title("Curva_de_reaccion_de_la_temperatura")
plt.ylabel("Temperatura_(C)")
plt.xlabel("Muestras(10s/muestra)")
plt.plot(sr,
         label= "Senial_del_Sistema")
plt.plot(shat,
         label= "Estimacion_Kalman",
         color="blue")
plt.plot(recta_tangente,
         color="purple",
         label="Tangente")
plt.axvline(6.02774,
            color="red",
```

```
            linestyle="dashed")
plt.axvline(51.607189,
            color="red",
            linestyle="dashed")
plt.axhline(y=21.235,
            color="g")
plt.axhline(y=26.273,
            color="g")
plt.legend()
plt.show()
```

Anexo C: Script que plotea el efecto del filtro de Kalman.

C.1. Programa en Matlab

```
A = [1.1269   -0.4940    0.1129
     1.0000         0         0
          0    1.0000         0];

B = [-0.3832
      0.5919
      0.5191];

C = [1 0 0];

D = 0;
Ts = -1;
sys = ss(A,[B B],C,D,Ts,'InputName',{'u' 'w'},'OutputName','y');  %
    Dinamica de la planta y adicion de ruido w
Q = 2.3;
R = 1;

[kalmf,L,~,Mx,Z] = kalman(sys,Q,R);
kalmf = kalmf(1,:);
sys.InputName = {'u','w'};
sys.OutputName = {'yt'};
vIn = sumblk('y=yt+v');

kalmf.InputName = {'u','y'};
kalmf.OutputName = 'ye';

SimModel = connect(sys,vIn,kalmf,{'u','w','v'},{'yt','ye'});
t = (0:160)';
u = sin(t/3);
rng(10,'twister');
w = sqrt(Q)*randn(length(t),1);
v = sqrt(R)*randn(length(t),1);
out = lsim(SimModel,[u,w,v]);
yt = out(:,1);    % Respuesta real
ye = out(:,2);  % Respuesta filtrada
y = yt + v;      % Mediciones
clf
```

```
subplot(211), plot(t,yt,'b',t,ye,'r--'),
xlabel('Muestras', 'FontSize', 18),
        ylabel('Salida', 'FontSize', 18)
title('Respuesta del filtro de Kalman', 'FontSize', 20)
legend('Real','Filtrada', 'FontSize', 20)
subplot(212), plot(t,yt-y,'g',t,yt-ye,'r--'),
xlabel('Muestras', 'FontSize', 18),
        ylabel('Error', 'FontSize', 18)
legend('Real(medida)','Real(filtrada)', 'FontSize', 20)
```

Anexo D: Script que plotea el efecto de los filtros digitales sobre una señal ruidosa.

D.1. Simulación de filtros digitales.

```
% Partiendo de la suposicion de tener las seniales 'x' (original) y
    'y' (ruidosa)

% Generacion de seniales de ejemplo
fs = 1000; % Frecuencia de muestreo
t = 0:1/fs:1;
x = sin(3*pi*50*t) + 0.5*sin(2*pi*120*t); % Senial original
y = x + 0.4*randn(size(t)); % Senial ruidosa

%% Filtro FIR
n_fir = 50; % Orden del filtro FIR
b_fir = fir1(n_fir, 0.7); % Filtro FIR pasa bajos
y_fir = filter(b_fir, 1, y); % Aplicacion del filtro FIR

%% Filtro IIR
[b_iir, a_iir] = butter(4, 0.7); % Filtro IIR Butterworth de orden
   4
y_iir = filter(b_iir, a_iir, y); % Aplicamos el filtro IIR

%% Filtro Wiener
y_wiener = wiener2(y, [5 5]); % Aplicamos del filtro Wiener

%% Filtro de Kalman
% Asumimos que el modelo de la senial es una senial lineal
A = 1; % Modelo de transicion del estado
H = 1; % Matriz de observacion
Q = 0.01; % Covarianza del proceso (ruido)
R = 0.1; % Covarianza de la medicion (ruido)
x_kalman = zeros(size(t)); % Inicializacion del estado estimado
P = 1; % Error de estimacion inicial
for k = 2:length(t)
    % Prediccion del estado
    x_pred = A * x_kalman(k-1);
    P_pred = A * P * A' + Q;

    % Ganancia de Kalman
    K = P_pred * H' / (H * P_pred * H' + R);

```

```
37      % Correccion
38      x_kalman(k) = x_pred + K * (y(k) - H * x_pred);
39      P = (1 - K * H) * P_pred;
40  end
41  y_kalman = x_kalman; % La senial filtrada es el estado estimado
42
43  %% Graficar resultados
44  figure;
45  subplot(6,1,1);
46  plot(t, x, 'g', 'LineWidth', 1.5); % Senial original
47  title('Senial_Original');
48  xlabel('Tiempo');
49  ylabel('Amplitud');
50  subplot(6,1,2);
51  plot(t, y, 'r', 'LineWidth', 1.5); % Senial ruidosa
52  title('Senial_Ruidosa');
53  xlabel('Tiempo');
54  ylabel('Amplitud');
55  subplot(6,1,3);
56  plot(t, y_fir, 'b', 'LineWidth', 1.5); % Senial filtrada con FIR
57  title('Senial_Filtrada_con_FIR');
58  xlabel('Tiempo');
59  ylabel('Amplitud');
60  subplot(6,1,4);
61  plot(t, y_iir, 'm', 'LineWidth', 1.5); % Senial filtrada con IIR
62  title('Senial_Filtrada_con_IIR');
63  xlabel('Tiempo');
64  ylabel('Amplitud');
65  subplot(6,1,5);
66  plot(t, y_wiener, 'c', 'LineWidth', 1.5); % Senial filtrada con
        Wiener
67  title('Senial_Filtrada_con_Wiener');
68  xlabel('Tiempo');
69  ylabel('Amplitud');
70  subplot(6,1,6);
71  plot(t, y_kalman, 'k', 'LineWidth', 1.5); % Senial filtrada con
        Kalman
72  title('Senial_Filtrada_con_Kalman');
73  xlabel('Tiempo');
74  ylabel('Amplitud');
```