Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería Mecánica Eléctrica

DISEÑO Y PROTOTIPO DE UN SISTEMA DE CONTROL DE LUMINISCENCIA Y RIEGO

Samuel Adolfo Choc Samayoa
Asesorado por el Ing. Walter Giovanni Álvarez Marroquín

Guatemala, noviembre de 2021

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

DISEÑO Y PROTOTIPO DE UN SISTEMA DE CONTROL DE LUMINISCENCIA Y RIEGO

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR

SAMUEL ADOLFO CHOC SAMAYOA
ASESORADO POR EL ING. WALTER GIOVANNI ÁLVAREZ MARROQUÍN

AL CONFERÍRSELE EL TÍTULO DE

INGENIERO EN ELECTRÓNICA

GUATEMALA, NOVIEMBRE DE 2021

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANA	Inga. Aurelia Anabela Córdova Estrada
VOCAL I	Ing. José Francisco Gómez Rivera
VOCAL II	Ing. Mario Renato Escobedo Martínez
VOCAL III	Ing. José Milton de León Bran
VOCAL IV	Br. Kevin Vladimir Armando Cruz Lorente
VOCAL V	Br. Fernando José Paz González
SECRETARIO	Ing. Hugo Humberto Rivera Pérez

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Pedro Antonio Aguilar Polanco
EXAMINADOR	Ing. Helmunt Federico Chicol Cabrera
EXAMINADORA	Inga. Ingrid Salome Rodríguez de Loukota
EXAMINADOR	Ing. Carlos Eduardo Guzmán Salazar
SECRETARIA	Inga. Lesbia Magalí Herrera López

HONORABLE TRIBUNAL EXAMINADOR

En cumplimiento con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación titulado:

DISEÑO Y PROTOTIPO DE UN SISTEMA DE CONTROL DE LUMINISCENCIA Y RIEGO

Tema que me fuera asignado por la Dirección de la Escuela de Ingeniería Mecánica Eléctrica, con fecha 3 de junio de 2019.

Samuel Adolfo Choc Samayoa

Guatemala, 26 de abril de 2021

Ing. Julio Solares
Coordinador del Área de Electrónica
Escuela de Ingeniería Mecánica Eléctrica
Facultad de Ingeniería, Universidad de San Carlos de Guatemala

Señor Coordinador:

Por este medio tengo el gusto de informar a usted, que he concluido con el asesoramiento del trabajo de graduación con título: **DISEÑO Y PROTOTIPO DE UN SISTEMA DE CONTROL DE LUMINISCENCIA Y RIEGO**, desarrollado por el estudiante *Samuel Adolfo Choc Samayoa,* con carné 201318619. Después de revisar su contenido final doy mi entera aprobación al mismo.

Atentamente.

Walter Giovanni Alvarez Marroquín
Ing. Electricista
colegiado No. 8113

Ing. Walter Giovanni Álvarez Marroquín

Colegiado No. 8113

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERIA

Guatemala, 3 de mayo de 2021

Señor Director
Armando Alonso Rivera Carrillo
Escuela de Ingeniería Mecánica Eléctrica
Facultad de Ingeniería, USAC

Estimado Señor director:

Por este medio me permito dar aprobación al Trabajo de Graduación titulado: **DISEÑO Y PROTOTIPO DE UN SISTEMA DE CONTROL DE LUMINISCENCIA Y RIEGO,** desarrollado por el estudiante **Samuel Adolfo Choc Samayoa**, ya que considero que cumple con los requisitos establecidos.

Sin otro particular, aprovecho la oportunidad para saludarlo.

Atentamente,

ID Y ENSEÑAD A TODOS

Ing. Julio César Solares Peñate
Coordinador de Electrónica

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA MECÁNICA ELÉCTRICA
FACULTAD DE INGENIERÍA
UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

REF. EIME 157 2021.

El Director de la Escuela de Ingeniería Mecánica Eléctrica, después de conocer el dictamen del Asesor, con el Visto Bueno del Coordinador de Área, al trabajo de Graduación del estudiante; SAMUEL ADOLFO CHOC SAMAYOA titulado: DISEÑO Y PROTOTIPO DE UN SISTEMA DE CONTROL DE LUMINISCENCIA Y RIEGO, procede a la autorización del mismo.

Ing. Armando Alonso Rivera Carrillo

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DIRECCION ESCUELA
RIA
MEC
CTRICA
INGENIERIA

GUATEMALA, 2 DE NOVIEMBRE 2,021.

Decanato
Facultad de Ingeniería
24189101 – 24189102
secretariadecanato@ingenieria.usac.edu.gt

DTG. 606-2021.

La Decana de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería Mecánica Eléctrica, al Trabajo de Graduación titulado: **DISEÑO Y PROTOTIPO DE UN SISTEMA DE CONTROL DE LUMINISCENCIA Y RIEGO**, presentado por el estudiante universitario: **Samuel Adolfo Choc Samayoa,** y después de haber culminado las revisiones previas bajo la responsabilidad de las instancias correspondientes, autoriza la impresión del mismo.

IMPRÍMASE:

Inga. Anabela Cordova Estrada
Decana

Guatemala, noviembre de 2021

AACE/asga

ACTO QUE DEDICO A:

Dios	Por permitirme vivir las experiencias necesarias para alcanzar este logro.
Mis padres	Amanda Samayoa Aldana de Choc y Perfecto Samuel Choc Guzmán, por haberme dado su apoyo y la educación apropiada.
Mis hermanos	Por apoyarme y darme consejos y palabras de motivación durante esta faena.
Mis amigos	Compañeros de clase universitaria, excompañeros de diversificado y hermanos en Cristo, por haber estado presentes en momentos críticos de la carrera.

AGRADECIMIENTOS A:

Universidad de San Carlos de Guatemala	Por dar la oportunidad de seguirse superando y continuar desarrollándose a los guatemaltecos que buscan crecimiento académico y profesional.
Facultad de Ingeniería	Por compartir sus instalaciones y haberme permitido desarrollarme como persona y como estudiante.
Departamento de Física	Por haberme aceptado como auxiliar, fue la oportunidad que necesite para continuar mis estudios.
Mi asesor	Ingeniero Walter Álvarez, por apoyarme en un momento definitivo para mi carrera, siempre le recordare agradecido y con aprecio.
Mis amigos	Cada uno por nombre, por ayudarme a mantener mi salud mental y recordarme siempre que podía contar con ellos.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

FIGURAS

TABLAS

LISTA DE SÍMBOLOS

Símbolo	Significado
I2C	Circuito inter integrado
ADC	Convertidor análogo a digital
IDE	Entorno de desarrollo integrado
Lux	Lumen por metro cuadrado
Ω	Ohmios
LDR	Resistor dependiente de luz
V	Voltio
W	Watts

GLOSARIO

Acople	Unión entre dos o más circuitos.
Actuador	Dispositivo que realiza una acción o movimiento al ser activado.
Amplificador	Dispositivo que aumenta la amplitud de una señal.
Analógica	Tiene un funcionamiento dependiente de fenómenos electromagnéticos.
Atenuación	Reducción de potencia de una señal.
BIT	Unidad mínima de información en comunicaciones informáticas.
Buzzer	Dispositivo utilizado para emitir alertas audibles.
Capacitancia	Capacidad de almacenar carga en forma de campo eléctrico.
Ciclo de trabajo	Es el porcentaje de un periodo en el que hay funcionamiento.
Circuito integrado	Dispositivo semiconductor de 5 bloques o más.

Codificar	Proceso de cambiar la forma de presentar información digital.
Comunicación Serial	Forma de comunicación en la que los bits van uno detrás del otro.
Condensación	Paso de un material de gaseoso a líquido.
Condicional	Es una sentencia que revisa si algo se cumple o no.
Electrodo	Extremo de un conductor en contacto con un medio.
Espectro	Dominio en que se distribuyen las ondas de acuerdo a su frecuencia.
Estomática	Relacionado a las dos células oclusivas que forman parte de la epidermis de la planta.
Fotosíntesis	Es el proceso metabólico de las plantas.
Frecuencia	Ciclos que realiza una señal por segundo.
Librería	Archivo que contiene algoritmos o declaraciones genéricas de uso repetitivo.
Lumen	Unidad de medida de flujo luminoso.
Luminiscencia	Es cualquier emisión de luz emitida por cualquier cuerpo sin importar su temperatura.

Metabolismo	Conjunto de cambios químicos y biológicos que se producen continuamente en las células.
Microcontrolador	Dispositivo que realiza una tarea determinada o la establece por medio de un conjunto de operaciones lógicas y señales.
Minicomputador	Computador de baja robustez de tamaño reducido.
Módulo	Es un sistema auxiliar para una tarea específica.
Ohmios	Unidad de medida de resistencia.
PCB	Placa de circuito impreso, (Printed circuit board). Soporte mecánico y eléctrico que conecta todos los componentes electrónicos por medio de hojas de cobre grabadas.
Periférico	Aparato o dispositivo auxiliar conectado a la unidad central.
Prototipo	Versión experimental de un dispositivo o aparato para investigación o promoción.
Puerto	Circuito físico a través del que se envían o reciben señales.
Resistencia	Oposición al paso de corriente eléctrica.

Resistor	Dispositivo pasivo lineal que posee resistencia.
Riego	Aplicación artificial de agua a las plantas por medio del suelo.
Señal	Variación de una magnitud que contiene información.
Traslúcido	Característica de los materiales que permiten el paso de la luz.
Umbral	Mínimo que debe tener una magnitud para ser detectada o incluida.
Variable	Símbolo utilizado para representar una cantidad sujeta a cambiar.
Voltaje	Diferencia de potencial eléctrico.
Voltios	Unidad de medida para voltaje.
Watts	Unidad de medida de potencia.

RESUMEN

Este documento contiene el proceso de diseño de un prototipo de sistema de control de luminiscencia y riego, se detallan la teoría, funcionamiento del software, apariencia de la interfaz de usuario, hardware, conexión del hardware, funcionamiento del hardware, diagramas esquemáticos y su funcionamiento.

En el primer capítulo contiene el marco teórico utilizado para definir las variables de interés, sus características e importancia en el proyecto, también incluye una descripción del funcionamiento del algoritmo de control y las señales de salida que tendrá el sistema.

El segundo capítulo detalla el software del sistema, incluye segmentos de código de las diferentes tareas que realizan el minicomputador y el microcontrolador, algoritmos de medición, el proceso aplicado a las mediciones para utilizar en el sistema de control y también muestra la presentación y funcionamiento de la interfaz de usuario.

El tercer capítulo caracteriza los sensores utilizados, incluye la conexión de estos y una explicación del funcionamiento, los actuadores y el principio físico que utilizan, las PCB de acople entre los sensores y el microcontrolador, y de acople entre actuadores y el minicomputador, y fotografías de estas partes del prototipo.

El cuarto capítulo muestra fotografías de todo el sistema en funcionamiento, con descripciones de lo que se puede visualizar en cada una de ellas.

OBJETIVOS

General

Diseñar el prototipo de un sistema de control de luminiscencia y riego.

Específicos

1. Caracterizar los sensores que se usaran para el control de la luminiscencia y el condensador de agua.

2. Diseñar una interfaz de usuario con opciones intuitivas.

3. Diseñar un registro de los datos del sistema para realizar análisis de los mismos.

INTRODUCCIÓN

El riego y la luz solar son fundamentales para el desarrollo estable en la agricultura, para tener control de las dosis de cada uno, es necesario intervenir durante el día a diferentes horas, esto depende de la temporada, el clima y otros factores externos que pueden afectar el desarrollo óptimo de cualquier planta en la agricultura, automatizar el proceso disminuye la necesidad de intervención humana y optimiza el tiempo de crecimiento como también el uso de recursos.

Como alternativa se ha elaborado un prototipo con el propósito de realizar las mediciones de luminiscencia y riego y finalmente tomar decisiones de riego o proteger el cultivo de la luz solar si así lo dispone.

Para el prototipo se ha utilizado un minicomputador Raspberry Pi3b+ como núcleo central del sistema, que se encargara de recibir las mediciones y procesarlas, la tensión de agua en el suelo se mide utilizando el método sugerido por el fabricante del sensor 200ss.

Este diseño cumple los requisitos para operar de manera constante ajustado con parámetros promedio, para utilizarse en cultivos específicos es necesario proveer al sistema de la configuración apropiada.

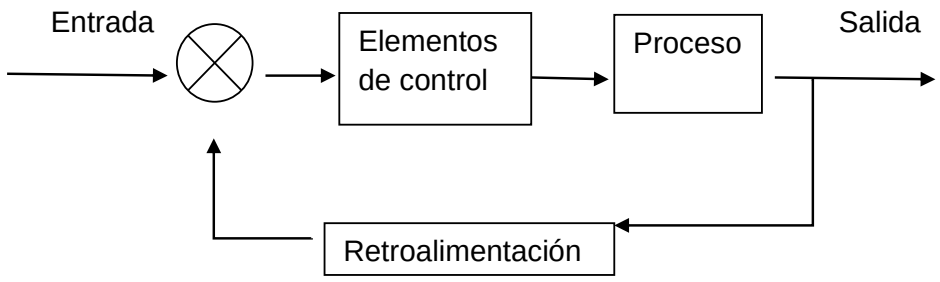
1. SISTEMA DE CONTROL DEL PROTOTIPO

Un sistema es un conjunto complejo de partes, elementos y (o) componentes que interactúan entre sí con al menos otro componente. El prototipo controla la luminiscencia y el riego, para tomar decisiones considera: fase de crecimiento, características del ambiente y lecturas de agua en el suelo.[1]

1.1. Sistema de lazo cerrado

Un sistema de lazo cerrado recibe señales externas o variables de entrada y tiene respuestas a las mismas llamadas variables de salida, estas dependen de: variables de control, perturbaciones en las variables de entrada y de la retroalimentación.

Figura 1. **Diagrama de bloques de un sistema de lazo cerrado**



Fuente: elaboración propia.

[1] DIAZ, Wilfredo. *Sistema de control.* https://es.slideshare.net/wilfredodiaz2/sistemas-de-control-50453873. Consulta: 13 de julio de 2015.

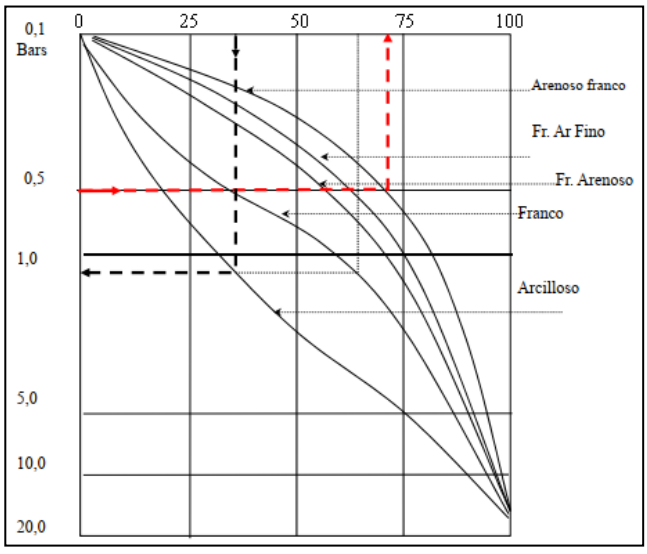
1.1.1. Señales de entrada

Es una variable del sistema controlado que se elige de modo tal que mediante su manipulación se logra que el sistema cumpla un objetivo determinado.

1.1.1.1. Umbral de riego

Con base en la fase de crecimiento se establecen límites en los que una planta se desarrolla con éxito, cuando realiza la fotosíntesis ocurre una apertura estomática que ocasiona pérdida de agua, debe reabastecerse absorbiéndola del suelo, si no consigue la suficiente se deshidrata y disminuye la tasa fotosintética, si la reduce por mucho tiempo se marchita.

Figura 2. **Curva tensión agotamiento del agua disponible**



Fuente: Facultad de Agronomía. *Agua en el suelo.* https://www.yumpu.com/es/document/view/14700910/agua-en-el-suelo-facultad-de-agronomia. Consulta: 4 de agosto de 2019.

El valor de esta variable depende únicamente de la etapa de desarrollo y el tipo de suelo usado, al no ser modificable por el sistema se considera como constante en cada periodo, debe considerarse que un exceso de riego también es dañino, por esta razón tanto la escases o exceso por periodos largos activaran una alerta indicando que requiere intervención del usuario.

Tabla I. **Potencial de matriz (succión) al cual se debería regar para obtener las productividades máximas**

Cultivo	Potencial de matriz(bar)	Referencia bibliográfica
HORTALIZAS		
Repollo	- (0.6 - 0.7)	Vittum et al, 1963; Pew, 1958
Arveja	- (0.3 - 0.5)	Taylor, 1965
Lechuga	- (0.4 - 0.6)	Vissar, 1959; Pew, 1958
Maiz dulce	- (0.5 - 1.0)	Vittum et al, 1963
Cebolla	- (0.45 - 0.65)	Pew, 1958
Papa	- (0.3 - 0.5)	Vittum et al, 1963; Pew, 1958
Zanahoria	- (0.3 - 0.5)	Pew, 1958
Tomate	- (0.8 - 1.5)	Vittum et al, 1963
Cebolla (semilla)	-1.5	Haise & Hagan, 1967
GRANOS		
Maíz(vegetativo)	- 0.50	Taylor, 1965
Maíz(maduración)	-8 -12	Taylor, 1965
Poroto	- (0.35 - 2)	Vittum et al, 1963
FRUTAS Y FRUTALES		
Frutilla	- (0.2- 0.3)	Haise & Hagan, 1967
Melón	- (0.35 - 0.40)	Marsh, 1961:; Pew, 1958
Limonero	-4	Haise & Hagan, 1967
Naranja	- (0.2-1.0)	Stolzy et al., 1963
FORRAJERAS		
Alfalfa	-1.5	Taylor, 1965
Trébol	-0.3 -0.8	Doorenbos & Pruitt, 1975
CULTIVOS INDUSTRIALES		
Caña de azúcar	- (0.8-1.5)	Doorenbos & Pruitt, 1975
Algodón	- (1 - 2)	Doorenbos & Pruitt, 1975

Fuente: Facultad de Agronomía. *Agua en el suelo.* https://www.yumpu.com/es/document/view/14700910/agua-en-el-suelo-facultad-de-agronomia. Consulta: 4 de agosto de 2019.

La línea roja punteada indica el límite de tensión recomendado en 0,5 bar que equivale a 50 kPa, este valor debe tomarse como referencia, la mayoría de los cultivos presentan el umbral de riego alrededor de este punto. Esta curva también muestra como un suelo arcilloso puede retener el agua mucho más que un suelo arenoso teniendo la misma cantidad de agua, he aquí la superioridad de medir la tensión de agua en el suelo comparada con la humedad relativa. Cabe aclarar que la tensión asciende hacia la parte inferior de la curva debido a que es una medida de succión.[2]

Las tensiones se presentan con valores negativos debido a que es una medida de succión, por fines prácticos se omitirá el signo y se tomará solo la magnitud de esta tensión en algunas secciones. Se presentan los potenciales de matriz, usados para fijar los umbrales de riego, de diferentes cultivos.

1.1.1.2. Luminiscencia

La luminiscencia también es una señal de entrada, cuando se suministra luz a una planta se deben tomar en cuenta tres características: La calidad de la luz, la duración y la cantidad de luz.

La calidad de la luz se refiere a las partes del espectro que se suministran, las plantas absorben la luz azul y la roja, el verde no se absorbe y por ello se ven verdes. La duración de exposición a la luz afecta en el tiempo de crecimiento, si se presentan días muy cortos el desarrollo puede verse ralentizado, este prototipo usa únicamente luz solar, por lo que se mencionan estos conceptos para ser tomados en cuenta en estudios posteriores.

---

[2] PACHÉS GINER, María. *El agua en el suelo: fuerzas de retención*. https://riunet.upv.es/bitstream/handle/10251/121154/Pach%C3%A9s%20-%20El%20agua%20en%20el%20suelo.%20Fuerzas%20de%20retenci%C3%B3n.pdf?sequence=1&isAllowed=y. Consulta: 24 de agosto de 2019.

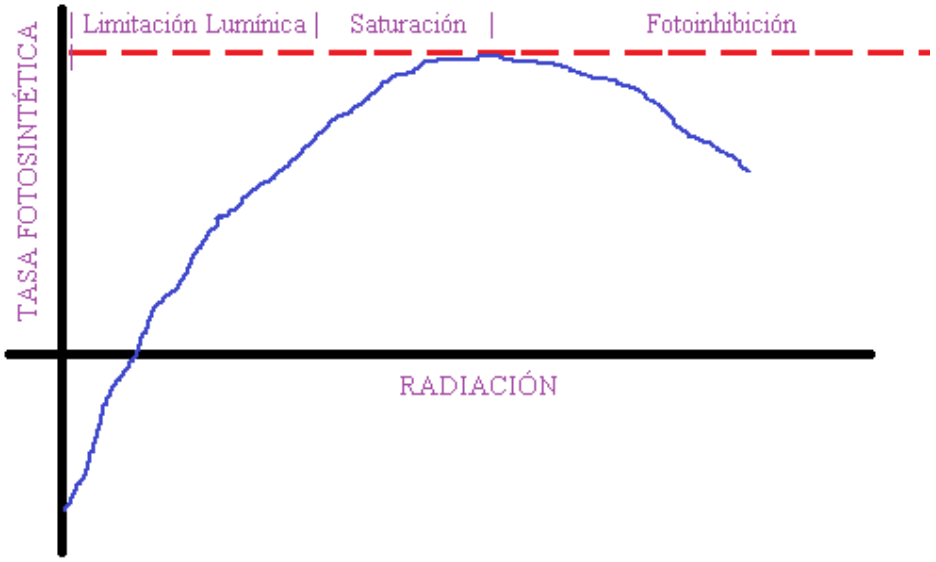
La cantidad de luz es la intensidad que recibe el cultivo, dependiendo de esta las plantas presentan tres regiones de comportamiento: limitación lumínica, saturación lumínica y fotoinhibición.

En limitación lumínica se usa más oxigeno del que se produce, la relación entre la intensidad y tasa fotosintética es lineal. En saturación lumínica se produce más oxigeno del que se usa, la relación entre la intensidad y tasa fotosintética no es lineal y se aproxima al punto de saturación que es la máxima tasa fotosintética sin sufrir daños. En fotoinhibición la radiación incidente es tan alta que la planta se protege reduciendo la absorción de luz y a la vez la tasa fotosintética.

> Clasificando las plantas por su metabolismo se obtienen tres tipos con diferente comportamiento a la incidencia de luz: C3 que son adecuadas para interiores, estas son más susceptibles a la exposición a la luz; C4 son adecuadas para exteriores, capaces de soportar exposición prolongada; y CAM adecuadas para zonas muy áridas, requieren mínima cantidad de agua.[3]

[3] infoAgro.com. *¿Tienen las plantas diferentes metabolismos?* https://www.infoagro.com/noticias/2019/_tienen_las_plantas_diferentes_metabolismos_.asp. Consulta: 26 de febrero de 2019.

Figura 3. **Curva de comportamiento tasa fotosintética, radiación**



Fuente: elaboración propia, empleando Windows Paint.

El sistema mantiene a la planta a un óptimo ritmo de desarrollo en la región de saturación lumínica, permitiendo la mayor cantidad de luz necesaria para mantener una alta tasa fotosintética, si por alguna razón se alcanza una posible fotoinhibición o deshidratación, se reduce la intensidad de luz con el actuador de sombra, este prototipo antepondrá mantener el cultivo a salvo, si por alguna razón se detecta ausencia de luz o exceso durante más de una hora, se activará una alerta indicando que requiere intervención humana.

1.1.1.3. Tiempo

Las alertas de exceso de riego, escases de riego, exceso de luz y escases de luz se activan si alguna de esas condiciones se mantiene por más de una

hora, en ese caso el sistema considera capacidad insuficiente debido a un clima extremo.

1.1.2. Elementos de control

Son utilizados para adecuar las señales de retroalimentación y señales de entrada, definir el objetivo cuantitativo que se busca alcanzar y hacer las mediciones de error correspondientes.

Los elementos de control de este prototipo consisten principalmente en condicionales que revisan comparaciones, se administra el inicio de funcionamiento, la luminiscencia y el riego.

1.1.2.1. Condiciones de funcionamiento

Para poder iniciar se deben cumplir cuatro condiciones: haberlo ordenado al sistema desde el botón en la interfaz de usuario, los datos de la planta deben estar guardados en el archivo todo.txt, debe tener suministro eléctrico constante y conexión exitosa con los sensores. Si alguna de estas no se consigue, se muestra un mensaje en la interfaz indicando la condición que hace falta, para la falta de suministro eléctrico se iniciará el apagado.

La revisión de los datos de la planta se realiza por medio de un método del programa, el suministro eléctrico se revisa utilizando una terminal digital de la Raspberry Pi 3 B+ conectada a un circuito de revisión y la conexión con los sensores se confirma por medio de software.

1.1.2.2. Condicional de riego

La activación del actuador de condensación se realiza al determinar que el suelo tiene menos agua disponible de la que el cultivo necesita, o la saturación de agua es menor al límite inferior recomendado para mantener la máxima productividad.

Se realiza la desactivación si se establece que el suelo alcanza la tensión recomendada para mantener la tasa fotosintética más alta.

El actuador de condensación utiliza un control proporcional integral, esto involucra una medición de error, la cual se determina por la diferencia entre la variable de entrada y la retroalimentación. El error del riego se determina por la cantidad de agua disponible actual y la cantidad de agua disponible ideal, el agua disponible actual se determina por medio de la tensión de agua en el suelo y el tipo de suelo. El agua disponible ideal se define por medio de la etapa de desarrollo.

1.1.2.3. Condicional de sombra

Inicialmente el prototipo fijara el actuador de sombra en nulo, en este modo toda la luz del ambiente incide en la planta, si la intensidad de la luz incidente es inofensiva el actuador continuara en este modo, después hay tres niveles de sombra, el primer nivel reducirá un 20 % de la luminiscencia, el segundo el 40 % del total y el último un 60 %, el sistema aumentará el nivel si detecta que la intensidad es muy alta o la saturación de agua está en el límite inferior recomendado, el sistema bajará de nivel si la intensidad es lo suficiente baja o la saturación de agua es suficiente para elevar la tasa fotosintética.

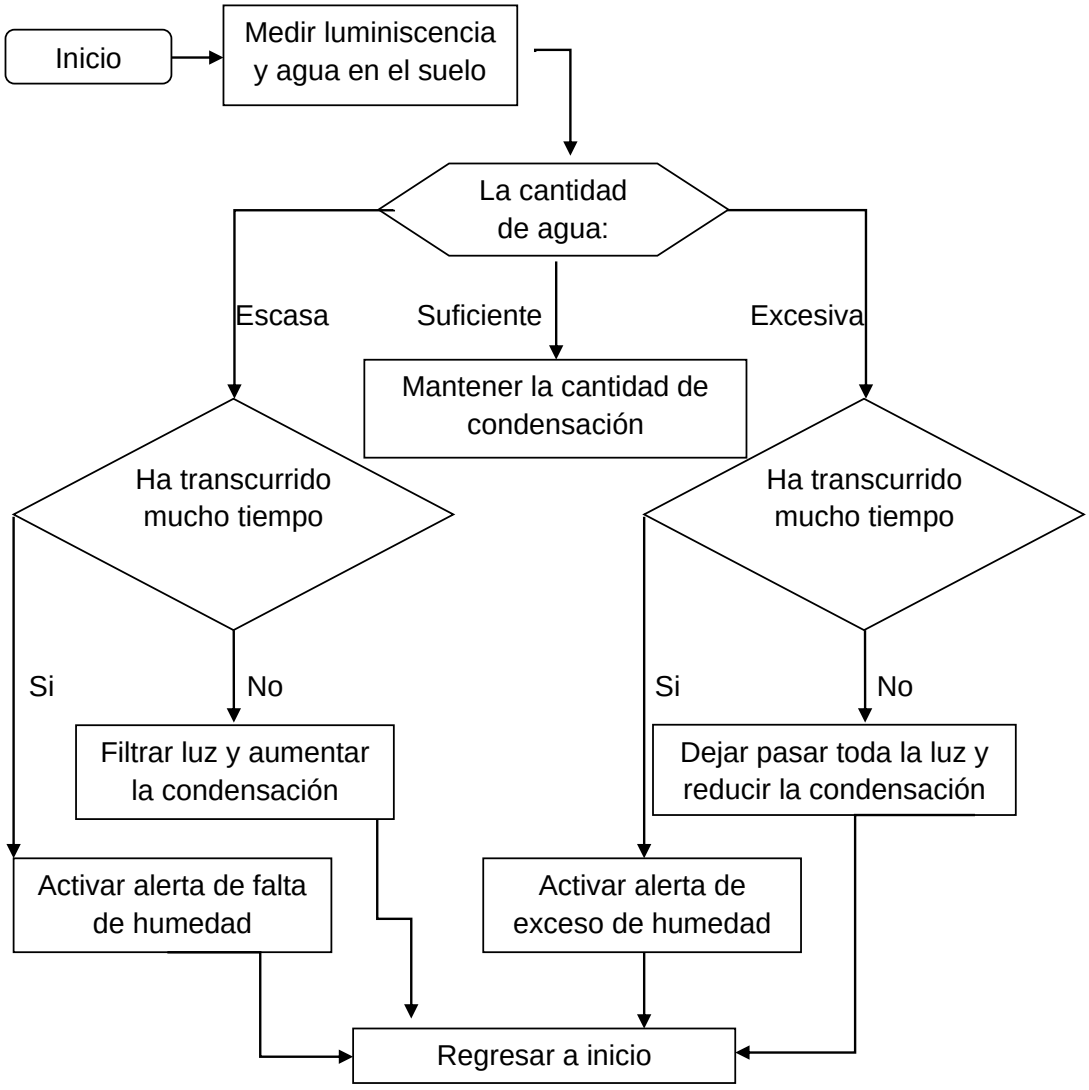
1.1.3. Proceso

El sistema incorpora ecuaciones para el control de su funcionamiento, que utiliza exclusivamente para definir las magnitudes de las variables de salida, estas variables se asignan a salidas hasta que el sistema es iniciado y son habilitadas las salidas, además de realizar estos cálculos ejecuta un algoritmo tanto para el riego como la luminiscencia.

El algoritmo de riego permite activar alertas y reconfigurar las variables de control cuando las condiciones del sistema cambian, este cambio es efectuado por la reducción de luz incidente que efectúa el actuador de sombra, al ser un cambio discreto de un minino de veinte por ciento, es lo más apropiado considerarlo en el control.

El algoritmo de luminiscencia también permite activar alertas y administra los niveles de sombra aplicados, los cambios de nivel dependen de la cantidad de agua que el cultivo tiene, además, verifica que los cambios se efectúan dentro de los niveles posibles.

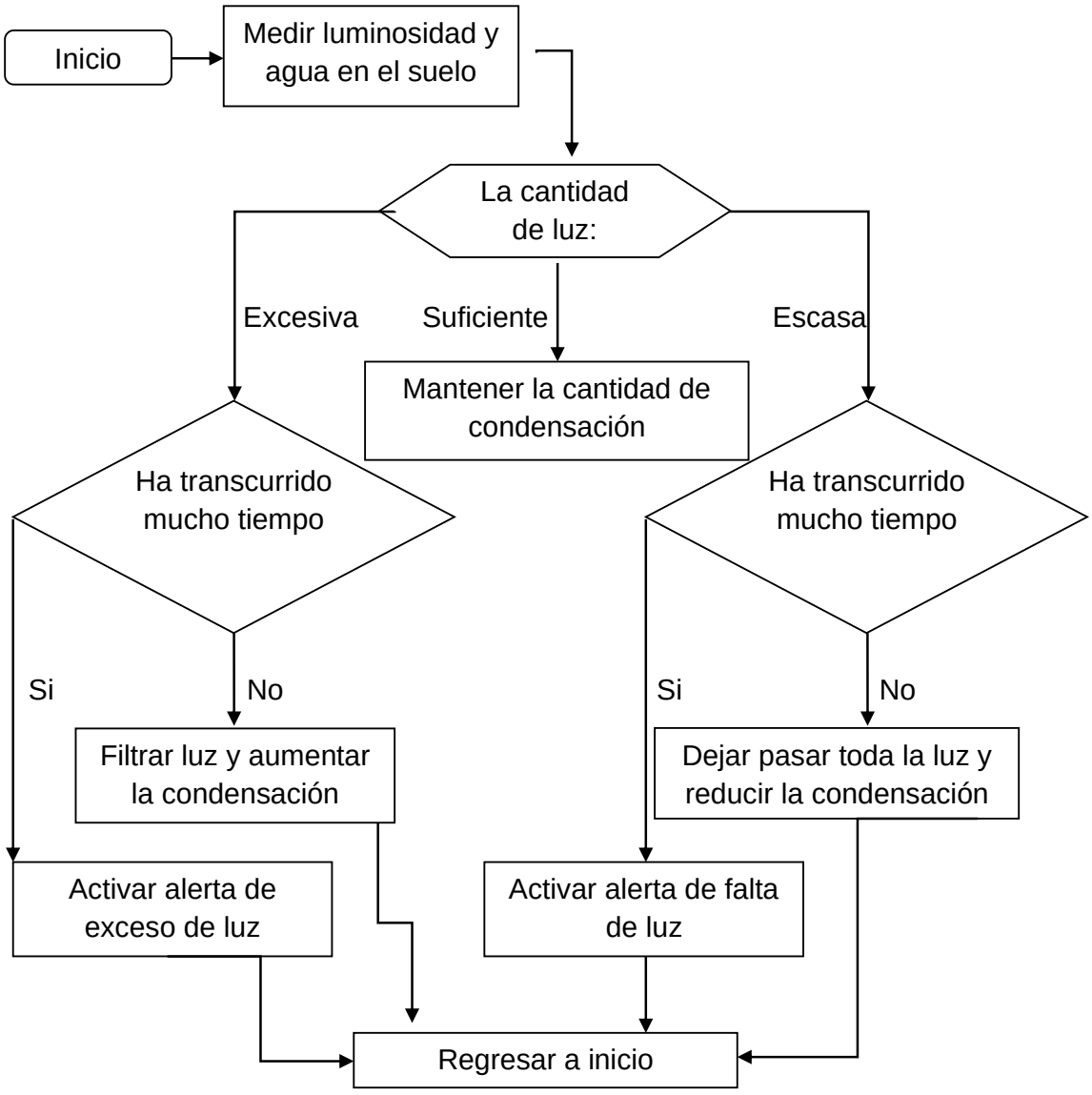
Figura 4. **Diagrama de flujo de la sección de riego**



Fuente: elaboración propia.

La manipulación de la luminiscencia en la sección de riego es una medida de precaución, mientras se realiza una correcta fotosíntesis se pierde agua, si se reduce la luz incidente la tasa fotosintética baja, de esta forma se protege al cultivo.

Figura 5. **Diagrama de flujo de la sección de luminiscencia**



Fuente: elaboración propia.

El diagrama de flujo del sistema indica la mutua dependencia entre el agua en el suelo y luminosidad, el ciclo se repite cada cinco segundos, las revisiones de escasez o exceso habilitan un conteo, si la condición se mantiene entonces

agrega uno, si cambia se reinicia el conteo, al llegar a sesenta se activa la alerta correspondiente.

1.1.4. Retroalimentación

La retroalimentación es una muestra de la variable de salida que se compara con la entrada, esta se usa para adecuar el proceso a lo requerido por el sistema.

La muestra de este prototipo es una medida de la tensión de agua en el suelo, indica que tan fácil es absorber el agua que hay en el suelo, esta tensión se compara con las tensiones límites del cultivo, esta información se usa en el control del actuador de condensación.

1.1.5. Señales de salida

La luminiscencia se controla por medio del actuador de sombra, la señal de salida es la posición del atenuador de luz o el desplazamiento requerido en cada ocasión, se suministra esta señal por medio de dos salidas digitales conectadas a un puente H, finalmente el puente H activa dos motores uno a la vez, que aumentan y reducen en nivel de sombra respectivamente.

El riego se controla por medio del actuador de condensación, el actuador consiste en una célula Peltier, la señal de salida es el ciclo de trabajo con el que la célula funciona, se trata de un PWM que administra la potencia de funcionamiento, a mayor ciclo de trabajo se consiguen mayores volúmenes de agua condensada, esta señal llega a un circuito de acople de potencia llamado en la sección 2 circuito de conmutación, este activa la célula Peltier.

El actuador de alertas es controlado con un puerto digital, únicamente se activa o desactiva por software, el resto de su funcionamiento es independiente.

1.1.5.1. Señal de desplazamiento

En el programa del prototipo se ha definido una función encargada de realizar los cálculos necesarios, para emitir las señales correspondientes a los actuadores. La señal de desplazamiento se determina con condicionales que revisan la tensión de agua en el suelo y el tiempo que ha transcurrido sin conseguir la cantidad de agua adecuada.

Figura 6. **Condicional de aumento de nivel de sombra**

```
if (self.c_m == 15 and self.pwm_condensador == 100 and self.n_s < 3):
    hilo4 = threading.Thread(target = clase.aumentarsombra, name='Condensador')
    hilo4.start()
    GPIO.output(self.salida_alarma, 0)
    self.c_m = 0
    self.bandera = 0
```

Fuente: elaboración propia, empleando Python 3.5.

Si han transcurrido 15 minutos con el máximo ciclo de trabajo y el nivel de luz es máximo, se realiza un aumento de nivel de sombra y se reinicia el contador, si ocurre de nuevo se repite el proceso, el prototipo cuenta con 3 niveles de sombra y un nivel sin ella, lo que da un total de 60 minutos para que el sistema emita la alerta de clima extremo. El proceso es el mismo para reducir nivel de sombra.

Figura 7. **Condicional de reducción de nivel de sombra**

```
if (self.c_m == 15 and self.pwm_condensador > 50 and self.n_s < 3 and self.l > self.l_max):
    hilo4 = threading.Thread(target = self.moversombra(36,40), name='Condensador')
    hilo4.start()
    GPIO.output(self.salida_alarma, 0)
    self.c_m = 0
    self.bandera = 0
    self.n_s += 1
```

Fuente: elaboración propia, empleando Python 3.5.

Los condicionales incluyen líneas de inhabilitación del actuador de alarmas y borrado del texto de alertas en el campo de la interfaz de usuario.

1.1.5.2. Señal de ciclo de trabajo

El ciclo de trabajo se determina por medio de una ecuación de control, en esta se revisa la diferencia entre la tensión actual y la tensión recomendable, dicha diferencia es ponderada por una constante proporcional, una integral y por el tiempo que ha transcurrido sin alcanzar la cantidad de agua recomendada.

Figura 8. **Ecuación de control y filtro de histéresis del ciclo de trabajo**

```
self.pwm_condensador = (self.c_p*(self.t_p - self.t_r))*(-1) + self.c_i*self.c_m
if self.pwm_condensador < 0:
    self.pwm_condensador = 0
elif self.pwm_condensador > 100:
    self.pwm_condensador = 100
```

Fuente: elaboración propia, empleando Python 3.5.

Posteriormente se aplica un filtro de histéresis, es un algoritmo que mantiene una variable dentro de un rango, en este caso el ciclo de trabajo no puede ser inferior a 0 %, y tampoco puede superar el 100 %.

2. SOFTWARE DEL SISTEMA

El prototipo utiliza el sistema operativo Raspbian, es libre y está basado en Debian, optimizado para el hardware de Raspbery Pi.

2.1. Control central

El cerebro del sistema es una Raspberry Pi3 modelo B+, es un minicomputador de arquitectura ARM, procesa los datos y toma todas las decisiones, el programa que utiliza el minicomputador para este prototipo está hecho en lenguaje Python en su versión 3.5.

2.1.1. Raspberry Pi 3

El programa que contiene la Raspberry Pi 3 incluye los algoritmos de medición, control y alertas del prototipo, además de la interfaz del usuario de la que se hablará en la sección 2.1.2. Luego de haber iniciado el sistema operativo, se abre el programa, a continuación, se ingresan los datos de la planta en cuestión o se busca en las almacenadas anteriormente, luego se pasa a funcionamiento al dar clic en iniciar sistema.

2.1.1.1. Uso del puerto GPIO

La administración de los puertos GPIO se hacen desde el IDE Python, se utiliza una librería llamada “RPi.GPIO” para configurar el uso y(o), salida que tiene cada puerto del minicomputador, el sistema utiliza salidas digitales, entradas digitales y periféricos de PWM.

Figura 9. **Ejemplo de configuración de puertos**

```
#Configuración de puertos

#Se importa la librería y se le da un nombre mas corto
import RPi.GPIO as GPIO

#indica que se llama a los puertos por su numero en la PCB
GPIO.setmode(GPIO.BOARD)

#Asigna el puerto 29 como salida, inicia en 0
GPIO.setup(29, GPIO.OUT, initial = 0)
```

Fuente: elaboración propia, empleando Python 3.5.

La sintaxis utilizada para la configuración de un puerto es: importar la librería “RPi.GPIO”, indicar el modo en que se llamaran los puertos (por su número en la PCB o por el número del pin del integrado al que están conectados), asignar al puerto (por el número correspondiente al modo elegido), el uso (entrada/salida) resistores pull-up o pull-down (de ser entrada) y el valor inicial (de ser salida).

Figura 10. **Ejemplo de configuración de periférico PWM**

```
#Configuración de periférico PWM

#Se importa la librería y se le da un nombre más corto
import RPi.GPIO as GPIO

#Indica que se llama a los puertos por su número en la PCB
GPIO.setmode(GPIO.BOARD)

#Asigna el puerto nombrado 35 como salida
GPIO.setup(35, GPIO.OUT)

#Se asigna el puerto y la frecuencia de salida del PWM
pwm1 = GPIO.PWM(35, 100)

#Se activa el periférico y se asigna el ciclo de trabajo inicial
pwm1.start(0)
```

Fuente: elaboración propia, empleando Python 3.5.

La sintaxis utilizada para la configuración de un periférico de PWM contiene algunos pasos más que la anterior, se debe agregar: indicar que esta terminal se conecta a un periférico PWM, fijar la frecuencia de funcionamiento del periférico e iniciar el periférico con un ciclo de trabajo elegido.

Si se desea, también es posible cambiar el ciclo de trabajo sin reiniciar el periférico.

Figura 11. **Sentencia de cambio de ciclo de trabajo**

```
#Cambio de ciclo de trabajo

#Se asigna un ciclo de trabajo a pwm1 de 90 que es 90%
pwm1.ChangeDutyCycle(90)
```

Fuente: elaboración propia, empleando Python 3.5.

Para desactivar todas las salidas y puertos (paso que se realiza cuando el sistema se suspende) se utiliza una instrucción simple que lo hace en un solo paso.

Figura 12. **Sentencia de desactivación total de puertos**

```
#Desactivación de todos los puertos

#Este paso desconecta todas las salidas
GPIO.cleanup()
```

Fuente: elaboración propia, empleando Python 3.5.

2.1.1.2. Adquisición de datos

El sistema realiza mediciones de luminiscencia, temperatura y tensión de agua en el suelo que obtiene por medio de un microcontrolador y mediciones de desplazamiento del actuador de sombra que recibe directamente en sus puertos digitales.

2.1.1.2.1. Datos recibidos por puerto serial

Entre el minicomputador y los sensores hay un microcontrolador como intermediario, este recibe los datos provenientes de los sensores por medio de protocolo I2C (para el sensor de luminiscencia), OneWire (para el sensor de temperatura), y lecturas analógicas (del sensor de matriz granular).

2.1.1.2.2. Medición de luminiscencia

La medición de luminiscencia se hace utilizando el sensor BH1750, este se conecta al microcontrolador el cual utiliza una librería para: realizar la solicitud de medición, recibir el paquete que contiene la información de la medición, decodificar dicho paquete y guardarlo en una variable que se usa posteriormente.

Figura 13. **Estructura del código para utilizar el sensor BH1750**

```
#include <BH1750.h> //Importa la librería del sensor
#include <Wire.h> //Importa la librería de comunicación
BH1750 Sensor_de_luz; //Se crea una instancia del sensor

float lux = 0.0; //Se crea una variable para las lecturas

void setup(){
  Serial.begin(57600); //Se inicializa la comunicación serial
  Sensor_de_luz.begin(); //Se inicializa la instancia del sensor
}

void loop(){
  lux = Sensor_de_luz.readLightLevel(); //Se realiza una lectura
  Serial.println(lux);//Envia la medicion de lumenes
  delay(1000); //Espera 1 segundo para la siguiente medición
}
```

Fuente: elaboración propia, empleando Python 3.5.

El programa para utilizar el sensor BH1750 debe contener los siguientes pasos: importar la librería para comunicarse con el sensor (Wire.h), importar la librería para inicializar la comunicación y procesar los datos del sensor (BH1750.h), crear una entidad que representa al sensor y enviarla a la librería BH1750.h para representar al sensor, inicializar la comunicación, guardar la medición del sensor en una variable y finalmente enviar la medida por el puerto serial.

2.1.1.2.3. Medición de temperatura

Para medir temperatura se utiliza el sensor DS18B20, se encuentra conectado al microcontrolador, requiere una librería para: realizar la solicitud de medición, recibir el paquete que contiene la información de la medición,

decodificar dicho paquete, pasarlo a unidades de la escala Fahrenheit o Celsius y guardarlo en una variable que se usa posteriormente.

Figura 14. **Estructura del código para utilizar el sensor DS18B20**

```
#include <OneWire.h>//Importa la librería de comunicación
#include <DallasTemperature.h>Importa la librería del sensor

const int pin_ds18b20 = 9;//Se crea una constante con el terminal del sensor
OneWire oneWireObjeto(pin_ds18b20);//Se inicializa la instancia de comunicación
DallasTemperature sensorDS18B20(&oneWireObjeto);//Se inicializa la instancia del sensor

void setup(){
  Serial.begin(57600);//Se inicializa la comunicación serial
  sensorDS18B20.begin();//Se inicializa la instancia del sensor
}

void loop(){
  sensorDS18B20.requestTemperatures();//Solicita la toma de mediciones
  Serial.println(lux);//Envia la medicion de lumenes
  delay(1000); //Espera 1 segundo para la siguiente medición
}
```

Fuente: elaboración propia.

Para utilizar el sensor DS18B20, el programa debe contener los siguientes pasos: importar la librería para comunicarse con el sensor (OneWire.h), importar la librería para inicializar la comunicación y procesar los datos del sensor (DallasTemperature.h), indicar a la librería OneWire.h la terminal donde se conecta el sensor en una entidad, enviar dicha entidad a la librería DallasTemperature.h para representar al sensor, inicializar la comunicación, solicitar la temperatura al sensor, guardar la medición del sensor en una variable y finalmente enviar la medida por el puerto serial.

2.1.1.2.4. Medición de tensión de agua en el suelo

Para medir la tensión de agua en el suelo se utiliza el sensor de matriz granular 200ss, se encuentra conectado al microcontrolador, no requiere de librerías, se realiza un algoritmo de polarizaciones y mediciones analógicas que se guardan en variables que se usan posteriormente.

Figura 15. **Estructura del código para sensor Watermark**

```
int medidal, medida2;

void setup(){
  Serial.begin(57600);
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(A0, INPUT);
}

void loop(){
  digitalWrite(2, LOW); //Polariza el sensor a 0v
  digitalWrite(3, HIGH);//Polariza el resistor a 5v
  delayMicroseconds(80);//Espera a que el voltaje en el sensor sea estable
  medidal = analogRead(A0);//Guarda el valor de voltaje entre el sensor y 0v
  digitalWrite(2, LOW);//Polariza todo el arreglo a 0v
  digitalWrite(3, LOW);
  delay(100);//Espera a que se descargue todo elemento
  digitalWrite(2, HIGH);//Polariza el sensor a 5v
  digitalWrite(3, LOW);//Polariza el resistor a 0v
  delayMicroseconds(80);//Espera a que el voltaje en el sensor sea estable
  medida2 = analogRead(A0);//Guarda el valor de voltaje entre el resistor y 0v
  digitalWrite(2, LOW);//Polariza todo el arreglo a 0v
  digitalWrite(3, LOW);
  delay(100);//Espera a que se descargue todo elemento

  Serial.println(medidal);//
  Serial.println(medida2);//
  delay(1000);
}
```

Fuente: elaboración propia, empleando Python 3.5.

El sensor de matriz granular se comporta como una resistencia variable, tiene una relación inversamente proporcional no lineal, esta medición de resistencia se realiza de forma indirecta, utilizando un divisor de voltaje compuesto por el sensor en serie con un resistor con resistencia conocida.

Este programa prescinde de librerías, el fabricante del sensor de matriz granular (Watermark), provee el algoritmo para utilizar estos sensores con un microcontrolador, los elementos que deben incluirse son: inicializar las terminales que controlan la alimentación del sensor, inicializar la terminal de lectura analógica del sensor, iniciar con cero voltios de alimentación al sensor, polarizar al sensor en un sentido, esperar que el voltaje en el sensor sea estable, realizar la lectura analógica del voltaje en el nodo del circuito y guardarla en una variable. Polarizar con cero voltios nuevamente, esperar que desaparezca cualquier voltaje por capacitancias parasitas, polarizar al sensor en el sentido opuesto al anterior, esperar que el voltaje en el sensor sea estable, realizar la lectura analógica del voltaje en el nodo del circuito y guardarla en otra variable, polarizar con cero voltios nuevamente y finalmente enviar estas mediciones por el puerto serial.

Debe notarse que el microcontrolador únicamente se encarga de realizar las lecturas y que estas aun necesitan ser procesadas para utilizarse en el sistema, cuando el minicomputador recibe estas medidas, debe procesarlas y convertirlas de lectura analógica a voltaje, de voltaje a resistencia y de resistencia a tensión de agua en el suelo.

2.1.1.2.5. Datos recibidos de forma directa

La administración del actuador de sombra pertenece al minicomputador, el se encuentra conectado directamente al puente H y los sensores de cambio de nivel del actuador de sombra.

2.1.1.2.6. Mediciones de nivel del actuador de sombra

El actuador de sombra consiste en dos rodillos con una banda de materiales traslúcidos, consta de varios segmentos para diferentes niveles de absorción de luz, cuando se pasa por completo de un nivel a otro un segmento de material opaco interrumpe un sensor de paso, la señal del sensor llega al minicomputador, esta señal le indica que hubo un cambio de nivel de sombra, al llegar al nivel deseado detiene el desplazamiento.

2.1.1.3. Procesamiento de datos

Como se mencionó al final de 2.1.1.2.4., el microcontrolador devuelve únicamente una lectura analógica de la medición de tensión de agua en el suelo, inicialmente esta medición debe convertirse en un voltaje, ya que usa un ADC de diez bits, se dividen los cinco voltios de operación del microcontrolador en 1023 niveles, lo que da un valor aproximado de 4,88 milivoltios por nivel, esta pequeña cifra debe multiplicarse por la lectura analógica, entonces el procedimiento para obtener el voltaje es:

$$V = \frac{L_a * V_{ref}}{2^n - 1}$$

Ec. 1.

Donde V es el voltaje presente en el sensor, L_a es la lectura analógica que obtiene el microcontrolador, V_{ref} es el voltaje de referencia del microcontrolador que divide en la cantidad de niveles del ADC y n es la cantidad de bits de resolución del ADC.

Para la segunda medición se debe realizar un ajuste antes de obtener el voltaje, ya que la medición se realiza en polarización inversa, el voltaje que hay en el sensor es V_{ref} menos el que se obtiene de la medición entonces para obtener Vel procedimiento cambia a:

$$V = V_{ref}\left(1 - \frac{L_a}{2^n - 1}\right)$$

Ec. 2.

Con ambas mediciones de V se calcula el promedio, ahora se procede a calcular la resistencia del sensor, al ser un divisor de voltaje formado por un resistor de valor conocido y el sensor, la ecuación para determinar la resistencia es:

$$R_s = \frac{R_0 V}{V_{ref} - V}$$

Ec. 3.

Donde R_s es la resistencia aparente del sensor, R_0 es el valor de resistencia conocida de un resistor fijo, V_{ref} es el voltaje de referencia del microcontrolador y V es el voltaje obtenido del promedio mencionado.

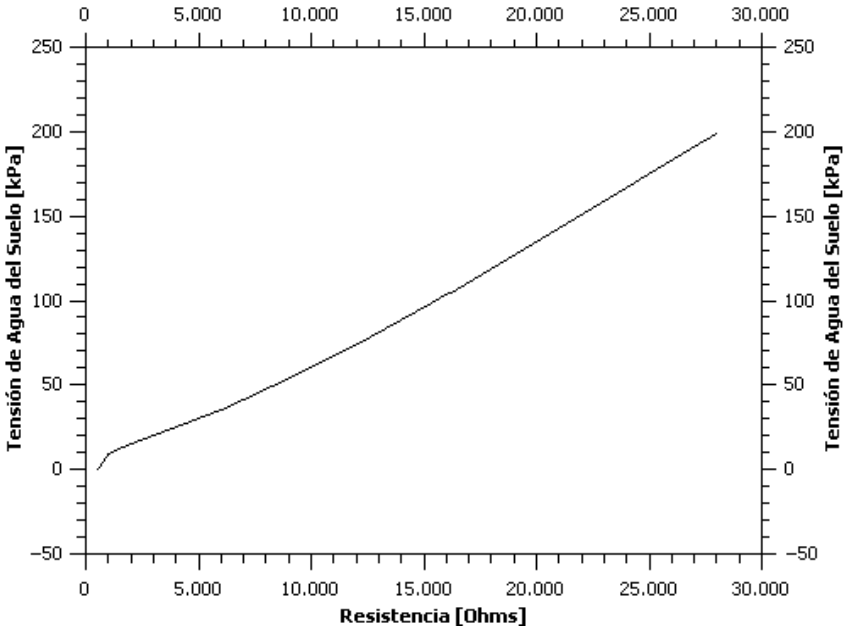
Para calcular la tensión también se necesita la temperatura del suelo, esta se obtiene de forma directa, el fabricante provee las ecuaciones para conocer la

tensión, cada una de ellas requiere la temperatura en Celsius y la resistencia en ohmios.

La figura 15 muestra el comportamiento no lineal de la relación entre tensión y resistencia del sensor, se elaboró con una serie de datos proveídos por el fabricante para calibración del monitor de medición, en un rango de 0 a 199 kPa con una temperatura constante de 24 ºC.

Para determinar la tensión de agua en el suelo se utilizan las mediciones de resistencia y de temperatura, se utilizan diferentes ecuaciones por ser la única forma de obtener una medida exacta de tensión, la primera es el límite inferior.

Figura 16. **Curva de comportamiento resistencia tensión**



Fuente: elaboración propia, empleando Qtiplot 0.9.8.9.

$$R_s < 550\ \Omega \quad T = 0$$

Donde R_s es la resistencia del sensor y Ω (omega mayuscula) representa la unidad de medida de resistencia ohmios, esta ecuación indica que para todo valor de resistencia menor a 550 ohmios la tensión es 0 kPa

$$550\Omega \leq R_s < 1\ 000\Omega$$

$$T = -20 * \left(\left(\frac{R_s}{1\ 000{,}00}\right) * \left(1{,}00 + 0{,}018 * (Temp - 24{,}00)\right) - 0{,}55\right)$$

Donde $Temp$ es la temperatura del suelo, esta ecuación se utiliza para el intervalo desde 550 ohmios hasta valores menores a 1000 ohmios, la tensión es negativa por ser una medida de succión.

$$1\ 000\Omega \leq R_s < 8\ 000\Omega$$

$$T = \frac{\left(-3{,}213 * \left(\frac{R_s}{1\ 000{,}00}\right) - 4{,}093\right)}{\left(1 - 0{,}009733 * \left(\frac{R_s}{1\ 000{,}00}\right) - 0{,}01205 * Temp\right)}$$

La ecuación se utiliza en el intervalo desde 1 000 ohmios hasta valores menores a 8 000 ohmios, en el siguiente intervalo tampoco se incluye 8 000 ohmios, esto debido a que dicho valor se encuentra en el punto intermedio entre ambos intervalos, el funcionamiento no se ve afectado por descartar este dato debido a que R_s nunca será exactamente igual a 8 000.

$$R_s > 8\ 000\ \Omega$$

$$T = -2{,}246 - 5{,}239 * \left(\frac{R_s}{1\,000{,}00}\right) * \left(1 + 0{,}018 * (Temp - 24)\right) - 0{,}06756$$
$$* \left(\frac{R_s}{1\,000{,}00}\right)^2 * \left(1{,}00 - 0{,}018 * (Temp - 24)\right)^2$$

La última ecuación se utiliza para cualquier dato de R_s mayor a 8 000 ohmios, los sensores están fabricados para utilizarse en el rango de 0 a 239 kPa, sin embargo, en la práctica las mediciones que superan los 100 kPa indican una dificultad de absorción extrema, dificultad que los cultivos no superan por lo que resulta de poca utilidad medir arriba de esta tensión.

2.1.1.4. Registro de datos

Cada planta guardada en el prototipo es almacenada en un archivo de texto llamado datos.txt, incluye sus datos generales: nombre, planta, familia, edad y etapa. También se almacenan las mediciones realizadas, en el archivo ultimamedida.txt la última medición y en otro llamado ultimoresultado.txt todas las que han sido tomadas desde el inicio de funcionamiento, los datos que se guardan en este último son: luminiscencia, tensión de agua en el suelo, temperatura, y número de alerta.

El primer registro sirve para realizar las búsquedas desde la interfaz de usuario, el segundo es un intermediario entre los hilos del programa, el tercero permite realizar depuración y revisión del comportamiento del prototipo.

2.1.1.5. Salida de señales

Luego de realizar la manipulación cuantitativa de las señales que se aplican a los actuadores, estas se suministran, todas las señales son determinadas por el sistema con base en constantes definidas en su elaboración.

El ciclo de trabajo del actuador de condensación se determina utilizando la diferencia entre la tensión recomendada y la actual, se pondera esta diferencia con la constante proporcional, también se agrega el tiempo en minutos que ha transcurrido sin alcanzar la tensión recomendada, y se pondera este tiempo con la constante integral.

$$PWM = -C_p(T_p - T_r) + C_i C_m(T_r - T_p)$$

Donde C_p es la constante proporcional, T_p es la tensión presente, T_r es la tensión recomendada, $C_i$es la constante integral, y C_m es el contador de minutos

El desplazamiento del actuador de sombra se activa al determinar que han transcurrido más de 15 minutos sin alcanzar la tensión recomendada, subir de nivel si la tensión es mayor y bajar si es menor, siempre se realiza un nivel a la vez.

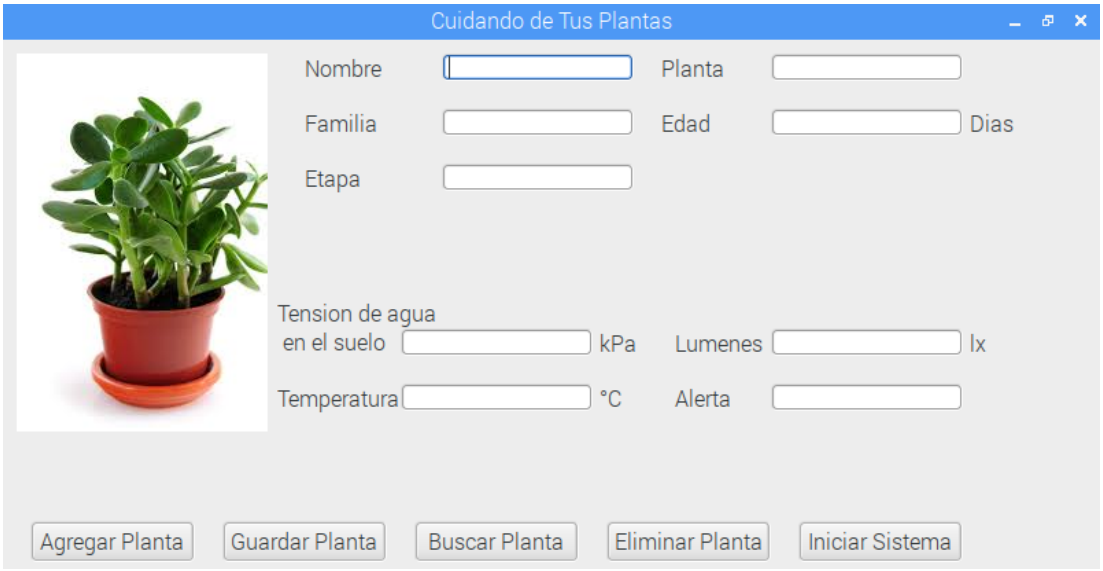
2.1.2. Interfaz del usuario

La interfaz del usuario se realiza a través de una pantalla táctil de 7 pulgadas y un teclado común, ambos dispositivos están conectados a la Raspberry Pi3. El software se desarrolló en el entorno python 3.5 utilizando las herramientas disponibles en PyQt.

La ventana de inicio contiene varios campos editables, los cinco de la parte superior contienen datos generales de la planta que puede ser nueva o guardada y los siguientes cuatro contienen los valores de las últimas mediciones realizadas y la alerta activa.

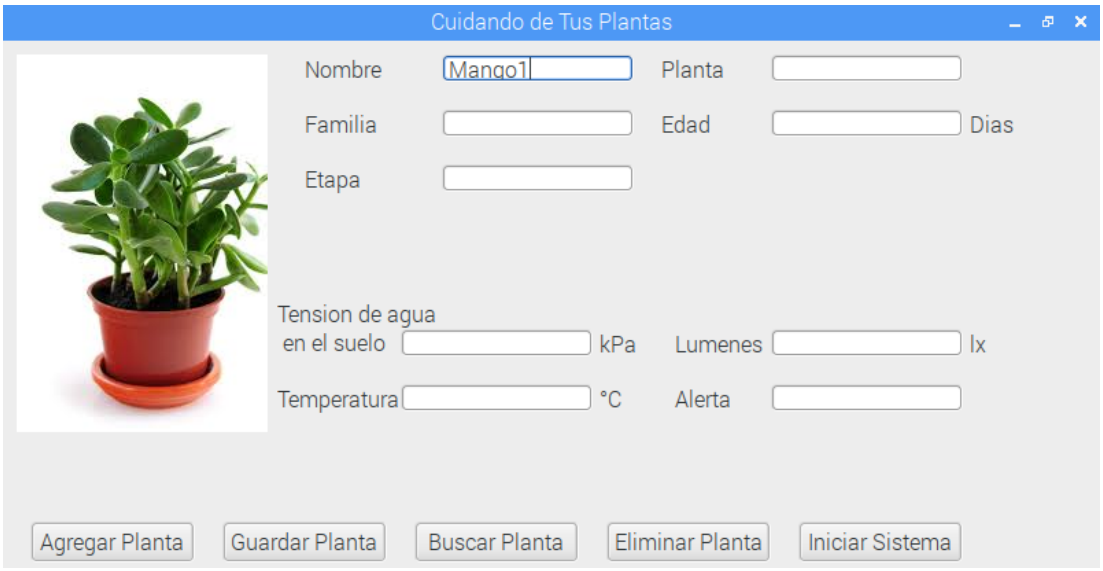
En la parte inferior posee cinco botones para realizar las siguientes acciones disponibles: agregar planta sirve para vaciar los campos fijos permitiendo colocar nuevos, guardar planta se utiliza para almacenar una nueva planta, buscar planta revisa si existe alguna almacenada con datos iguales y los muestra, eliminar planta borra la seleccionada e iniciar sistema que inicia el funcionamiento.

Figura 17. **Pantalla inicial de la interfaz de usuario**



Fuente: elaboración propia, empleando Python 3.5.

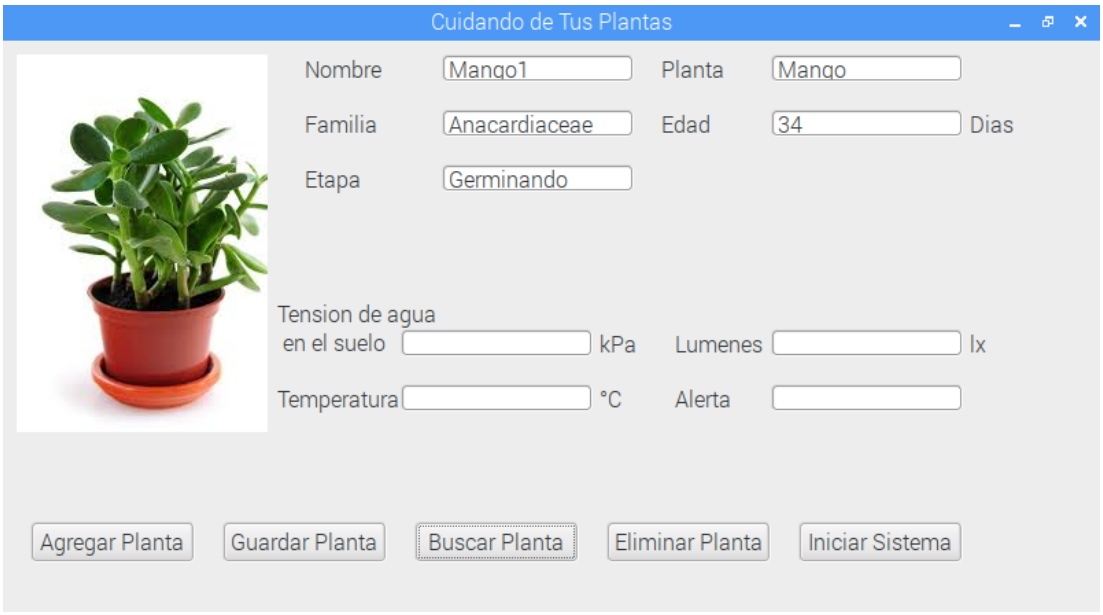
Figura 18. **Edición de campo de la interfaz de usuario**



Fuente: elaboración propia, empleando Python 3.5.

Aunque todos los campos son editables, las plantas guardadas solo se almacenan con los primeros cinco datos generales en el archivo datos.txt, los campos correspondientes a mediciones solo son modificados por el sistema cuando está en funcionamiento.

Figura 19. **Uso de buscar planta de la interfaz de usuario**



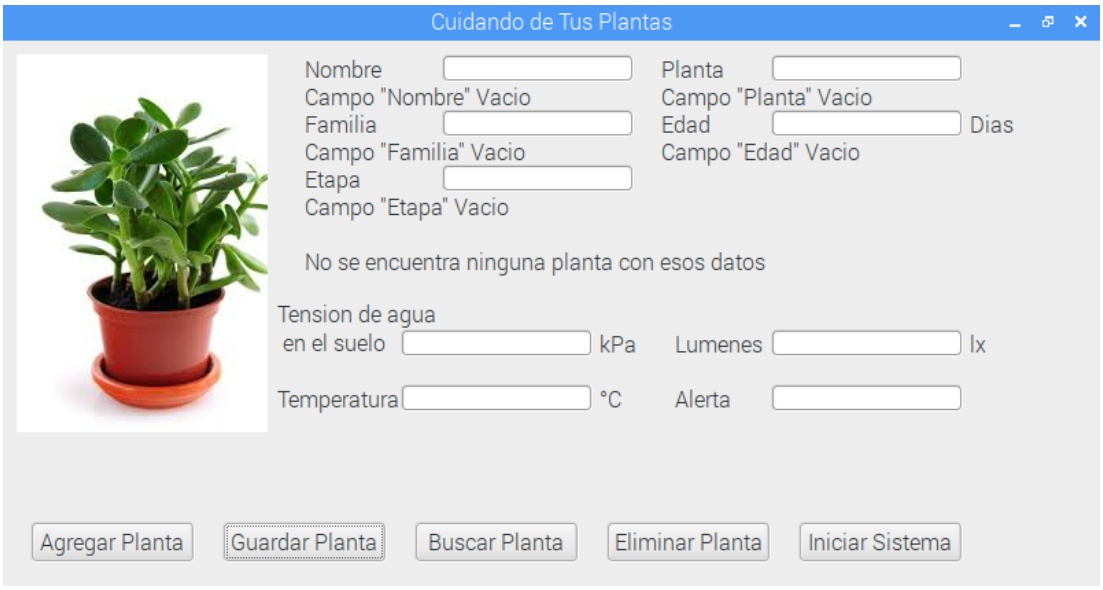
Fuente: elaboración propia, empleando Python 3.5.

En la figura 19 se ha presionado el botón buscar planta, dado que existe se muestran los demás datos generales que pertenecen a la misma, en ese momento se puede proceder a iniciar el sistema. Al realizar una búsqueda se requiere llenar al menos uno de los tres primeros campos correspondientes a: nombre, planta y familia.

Edad muestra la cantidad de días que la planta tiene de vida, este número se calcula de acuerdo con la fecha en que se guardó y la edad colocada inicialmente.

La interfaz también muestra advertencias en las cercanías de los campos editables que indican si hay campos vacíos, si no encuentra la planta que se busca, o si hay problemas para iniciar.

Figura 20. **Advertencias de campos de la interfaz de usuario**



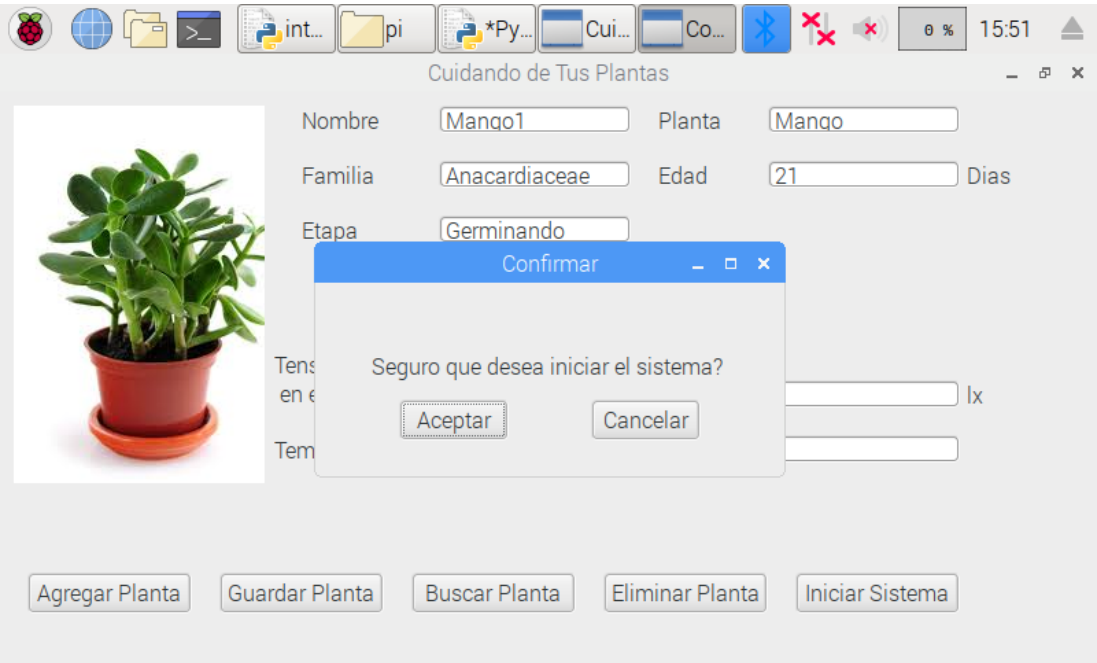
Fuente: elaboración propia, empleando Python 3.5.

En la figura 20 se ha presionado el botón buscar planta con todos los campos vacíos, se indican los campos que deben llenarse para realizar una búsqueda y se advierte que no se encuentra esa planta.

2.1.2.1. Inicialización

El sistema debe iniciarse manualmente presionando el botón de Iniciar Sistema que se encuentra en la esquina inferior derecha, tanto para un inicio normal como cuando el funcionamiento fue interrumpido de forma inesperada. Cuando el sistema se encuentra activo el botón cambia de nombre a Detener Sistema, y regresa al primer nombre cuando se ha detenido.

Figura 21. **Ventana de confirmación de la interfaz de usuario**



Fuente: elaboración propia, empleando Python 3.5.

En la figura 21 se ha presionado el botón iniciar sistema, al estar los campos llenos se realiza una confirmación para iniciar en una ventana emergente.

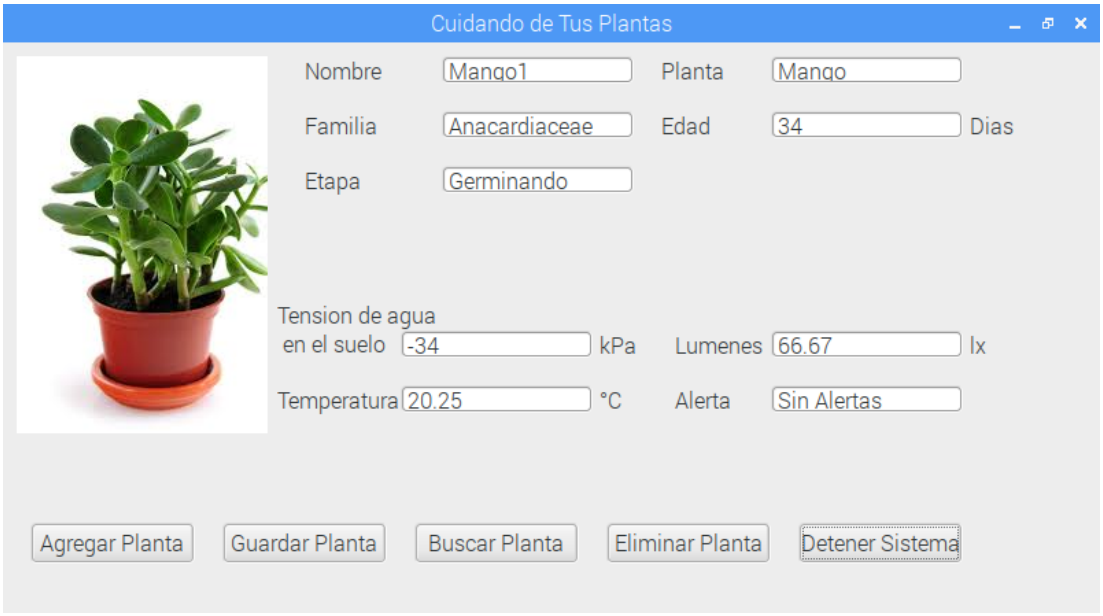
2.1.2.1.1. Condiciones de inicio

Al presionar el botón Iniciar Sistema se revisan las condiciones de inicio, estas incluyen: asegurar que se cuenta con información suficiente para iniciar, asegurar que hay comunicación con el módulo de periféricos y verificar funcionamiento normal en los sensores conectados al módulo de periféricos.

La primera condición consiste en revisar que los campos que corresponden a datos de la planta estén llenos, estos campos son: nombre, planta, familia y edad. Si falta al menos uno de estos datos se muestra una advertencia en cada campo que haga falta y el sistema no iniciará.

La segunda condición se revisa cuando se trata de abrir el puerto serial que conecta con el módulo de periféricos, si hay una excepción relacionada con este proceso, se muestra una advertencia indicando el problema y el sistema no inicia.

Figura 22. **Inicio exitoso del sistema**

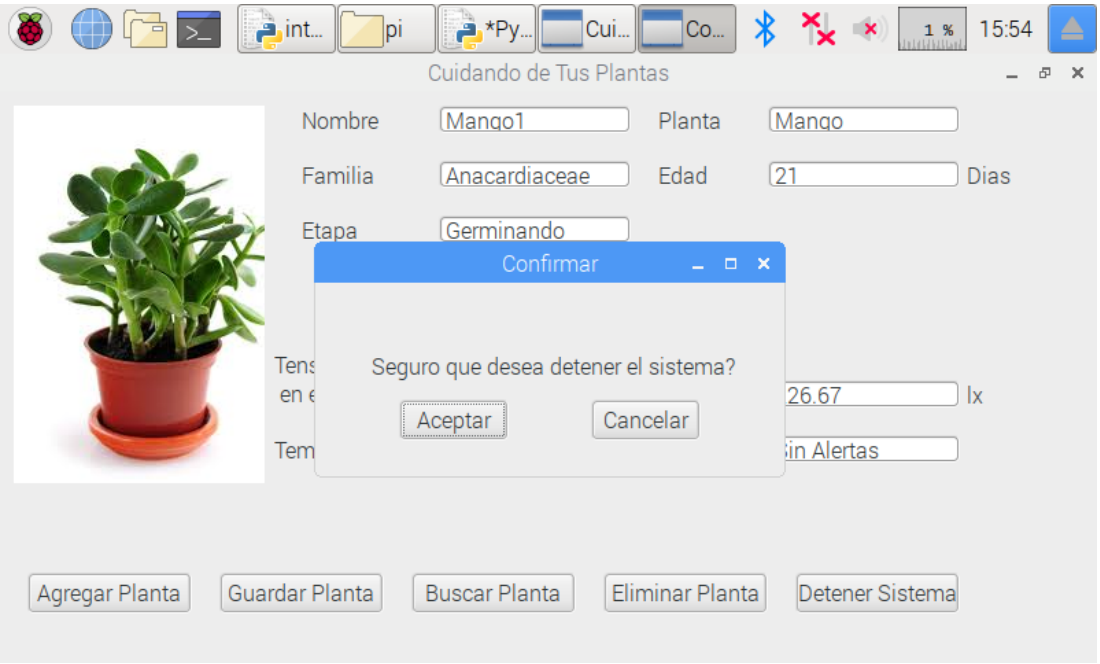


Fuente: elaboración propia, empleando Python 3.5.

Si los campos están llenos, se hace una prueba de comunicación serial, si hay conexión exitosa por el puerto serial, se realiza la revisión de sensores conectados al módulo de periféricos. Si la medición de temperatura es la medida por defecto indicando mala conexión con el sensor de temperatura se mostrará

una alerta, si hay un mensaje de error del sensor de luminiscencia por falla en la comunicación o el sensor de matriz granular indica circuito abierto, entonces se mostrará una advertencia indicando este inconveniente en el campo alerta.

Figura 23. **Confirmación de detención del sistema**



Fuente: elaboración propia, empleando Python 3.5.

2.1.2.2. Alertas

Las alertas de este prototipo tienen el propósito de indicar intervención directa del usuario debido a la detección de un comportamiento inesperado en el funcionamiento o una falla. Las alertas incorporadas incluyen avisar sobre: falta de suministro de corriente eléctrica, error en la comunicación con el módulo de periféricos, lecturas inesperadas en los sensores o condiciones del ambiente extremas.

La falta de suministro de corriente eléctrica se detecta por medio de un terminal de la Raspberry Pi3, al ser detectada la ausencia se activa una alerta sonora y se muestra en la interfaz de usuario la alerta, si se debe a un error el sistema espera 30 segundos para realizar la reconexión, de lo contrario se inicia la secuencia de apagado de seguridad. La secuencia consiste en detener el sistema y apagar la Raspberry Pi3 por medio de instrucciones del sistema.

Errores en la comunicación con el módulo de periféricos también pueden ocurrir por accidentes, si estando el sistema en funcionamiento se detecta esta falla, el sistema se detiene, activa una alerta sonora y muestra la alerta. Después de ocurrir este error debe volver a iniciarse el sistema manualmente desde la interfaz de usuario. Las lecturas inesperadas pueden deberse a que los sensores o conductores entre módulo de periféricos y sensores se han dañado, si se detecta esta falla estando en funcionamiento, el sistema se detiene, activa una alerta sonora y muestra la alerta. Después de ocurrir este error debe volver a iniciarse el sistema manualmente desde la interfaz de usuario.

Tabla II. **Alertas correspondientes a las fallas posibles**

Evento ocurrido	**Alerta mostrada**
Ausencia de suministro eléctrico	Falla de suministro
Falla de comunicación con el sensor de temperatura	Error DS18B20
Posible falla de conductores del sensor de matriz granular	Error Tensión
Funcionamiento correcto	Sin Alertas

Fuente: elaboración propia.

2.1.2.3. Registro de desarrollo

Todos los datos obtenidos por el sistema se almacenan en archivos de texto, en el prototipo se utilizan cinco archivos diferentes, el registro de desarrollo tiene el nombre que asigno a la planta al guardarse. Incluye los datos necesarios para realizar futuros estudios sobre el funcionamiento del proyecto, el comportamiento del clima a lo largo de su utilización y el de la planta con respecto a su intervención.

Cada línea en el registro de desarrollo contiene los datos mencionados separados por comas. El registro almacena datos generales como: nombre, planta, familia, edad y etapa correspondiente. También las mediciones de tensión de agua en el suelo, temperatura, luminiscencia, alerta, ciclo de trabajo del condensador, nivel de sombra, tensión de agua en el suelo anterior, tensión de agua en el suelo recomendada, contador de minutos, bandera de clima, variable de falla, hora y fecha.

Tensión anterior es la tensión de la medición anterior a la actual, tensión recomendada es la que el sistema busca alcanzar, bandera de clima es la variable que advierte si se requiere intervención del usuario, variable de falla indica la presencia de suministro eléctrico.

3. HARDWARE DEL SISTEMA

El prototipo cuenta con dispositivos de entrada utilizados para las mediciones y otros de salida que serán llamados actuadores en esta sección.

3.1. Mediciones del sistema

Las mediciones necesarias son agua en el suelo y luz en el ambiente, cada una utiliza dispositivos independientes de la otra.

3.1.1. Agua en el suelo

El prototipo determina la necesidad de riego midiendo la dificultad de la planta para absorber la que hay en el suelo, realizar dicha medición requiere de un sensor de temperatura y un sensor de matriz granular.

3.1.1.1. Medición de tensión de agua en el suelo

Comúnmente se utiliza la medición de humedad relativa para controlar el riego por medio de una sonda con electrodos descubiertos, esta medida proporciona información limitada debido a que omite si dicha agua es fácilmente absorbible por el cultivo, la medida de tensión en cambio, indica de forma directa si el agua es utilizable. Se cuenta con suficiente documentación sobre el comportamiento de la tensión del agua en el suelo dependiendo del tipo de suelo y el agua disponible para la planta.

3.1.1.1.1. Sensores de matriz granular

Consisten en dos electrodos de alta resistencia a la corrosión inmersos en yeso, el yeso se humedece dependiendo de la facilidad que hay de absorber el agua de alrededor, a menor magnitud de tensión (menor dificultad de absorción) su conductividad también aumenta y de esta forma se obtiene de forma indirecta la tensión de agua en el suelo midiendo la resistencia aparente en el sensor.

Figura 24. **Sensor 200ss**

Fuente: IRROMETER. *Medición de la humedad del suelo*. https://www.irrometer.com/sensorssp.html#wm. Consulta: 19 de enero de 2020.

Tabla III. **Características del sensor 200ss**

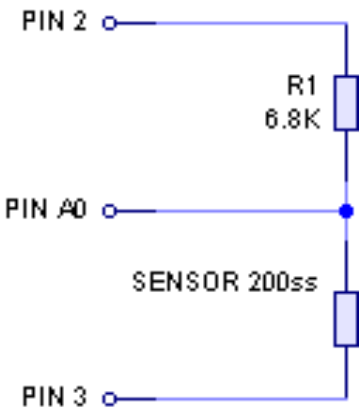
Rango de medición	0 a 239 cb
Circuito de lectura	AC y DC
Tiempo de muestra	10 s

Fuente: elaboración propia.

3.1.1.1.2. Circuito de medición

La resistencia aparente del sensor también es obtenida de forma indirecta, el circuito de medición es un divisor de voltaje, está formado por un resistor de valor conocido en este caso de 6,8 Kilo ohms en serie al sensor, los extremos del circuito son polarizados a 5 v y 0 v de forma alterna y en la unión de ambos dispositivos se realiza la medición del voltaje con una lectura analógica.

Figura 25. **Circuito de sensor de tensión**



Fuente: elaboración propia, empleando LiveWire 2004.

3.1.1.1.3. Sensor de temperatura LS18B20

Una propiedad intensiva de los materiales es la resistividad, esta característica depende de la temperatura del ambiente, el comportamiento de esta variación se utiliza en la medición de temperatura.

Es un sensor de temperatura encapsulado en una carcasa metálica, distribuido para usarse como sonda sumergible lo que permite medir temperatura en líquidos. Utiliza el protocolo de comunicación 1-wire, haciendo su conexión muy sencilla.

Tabla IV. **Características del sensor LS18B20**

Rango de temperatura	- 55 a 125 ºC
Resolución programable	9 a 12 bits
Precisión	0,07 ºC
Voltaje de alimentación	3 a 5,5 V
Corriente de alimentación	1 mA a 4 mA
Tiempo de muestra	93,75 a 750 ms

Fuente: elaboración propia.

3.1.1.1.4. Algoritmo de medición

La forma correcta de utilizar este sensor consta de una secuencia establecida por el fabricante: Inicialmente se polariza el circuito en un sentido que suele ser sensor a 0 v y resistor a 5 v; se espera por un corto periodo a que el voltaje sea estable descartando capacitancias parasitas, limitado para evitar la

corrosión de los electrodos; se toma la lectura analógica que equivale al voltaje presente en el sensor; se polariza a todo el circuito a 0 v nuevamente para descargar voltajes residuales; luego se polariza al circuito en el sentido contrario al del paso inicial y se repiten los pasos siguientes. El prototipo repite este procedimiento cada dos segundos y refresca con los últimos datos obtenidos los mostrados en la interfaz de usuario.

3.1.2. Luz en el ambiente

Todo cultivo necesita luz para sobrevivir, la intensidad de la misma debe alcanzar un mínimo que permitan un desarrollo adecuado y evitar excesos que lo marchiten. Las longitudes de onda que las plantas absorben son poco más que las incluidas en el espectro visible, por ello se utiliza un sensor de luminiscencia en este prototipo, este proporciona una medida de la intensidad de la luz visible, esta se mide en lux que equivale a un lumen por metro cuadrado.

3.1.2.1. Medición de luz ambiental

Para percibir la luz se utiliza una LDR o un fotodiodo, este último por medio del efecto fotoeléctrico devuelve un voltaje proporcional a la luminiscencia incidente, este voltaje es amplificado y luego se utiliza de acuerdo a lo que se desea hacer.

En el caso del BH1750 el fotodiodo utilizado tiene una respuesta muy parecida a la del ojo humano, el voltaje que proporciona se amplifica y se convierte de analógico a digital, luego es codificado y enviado de forma serial a un microcontrolador.

Figura 26. **Sensor BH1750**

Fuente: elaboración propia.

El sensor BH1750 se encuentra encapsulado en un módulo con terminales de conexión.

Tabla V. **Características del sensor BH1750**

Voltaje de alimentación	2.4 a 5V
Temperatura de operación	- 40 a 85 ºC
Temperatura de almacenamiento	- 40 a 100 ºC
Corriente de alimentación	120 a 190 µA
Protocolo de comunicación	I2C
Precisión en baja resolución	4 lx
Precisión en alta resolución	0,17 lx
Tiempo de muestra en baja resolución	16 a 24 ms
Tiempo de muestra en alta resolución	120 a 180 ms

Fuente: elaboración propia

Figura 27. **Ubicación del sensor BH1750**

Fuente: elaboración propia.

3.2. Actuadores del sistema

Este prototipo utiliza actuadores para controlar el riego, la intensidad de luz suministrada, alertas y el seguidor solar, la comunicación hacia los primeros dos dispositivos se realiza por medio de un acople de potencia, un circuito de conmutación para el actuador de riego y un puente H para el actuador de sombra, el tercero con una terminal de control y el cuarto funciona de forma independiente.

3.2.1. Actuador de riego

Tomando en cuenta que se puede prescindir de un suministro de agua, en este proyecto se obtiene por medio de condensación, el dispositivo encargado de esta tarea reduce la temperatura de un cono de aluminio que posteriormente destila en su superficie y derrama el agua en el suelo del cultivo.

Figura 28. **Actuador de riego**

Fuente: elaboración propia.

3.2.1.1. Punto de rocío

Se le da este nombre al efecto que se obtiene cuando se alcanzan las condiciones de humedad relativa y diferencia de temperaturas entre el aire y una superficie dada para condensar agua en dicha superficie.

Este efecto ocurre de una forma definida fija, las condiciones para lograrlo son relativas esto permite que se pueda conseguir por medio de frio y de calor, de forma simplificada la temperatura de rocío se calcula con la siguiente ecuación.

$$T_r = T_a + 35 * Log(\frac{H_r}{100})$$

Donde T_r es la temperatura de rocío, T_a es la temperatura del ambiente y H_r es la humedad relativa.

Figura 29. **Superficie de condensación**

Fuente: elaboración propia.

Un caso si se tapa un recipiente con contenido caliente o una salida de vapor con un objeto a temperatura ambiente, la superficie del objeto es fría en comparación con el ambiente, en este caso con el vapor que sale, la humedad relativa es alta y permite que se condense agua en la superficie del objeto.

El segundo caso se presenta cuando un objeto refrigerado se somete a la temperatura ambiente, la superficie del objeto es fría en comparación con el

ambiente, dependiendo de la magnitud de esta diferencia de temperaturas y la humedad relativa este objeto condensara más o menos agua en su superficie. El actuador de condensación funciona de acuerdo a este caso.

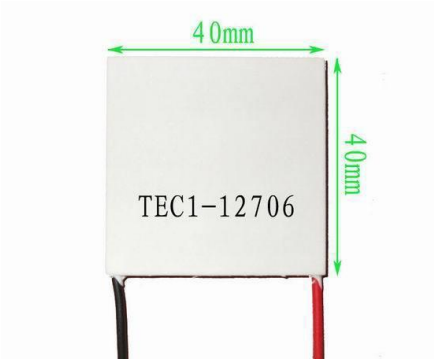
3.2.1.2. Placas Peltier

Una placa o célula Peltier es un dispositivo formado por dos superficies y una estructura interna de bloques N y P, cuando una corriente eléctrica circula por este arreglo los electrones que consiguen atravesar la barrera de potencial son los que poseen mayor energía, los mismos que poseen mayor calor, al perder esos electrones ese lado se enfría y el otro se calienta.

Las placas Peltier son fabricadas con Teluro y Bismuto, materiales con buena conductividad eléctrica y mala conductividad térmica, lo que reduce el efecto joule y mejora la absorción de calor en el lado frío.

Para utilizar estos dispositivos se requiere de un circuito que administre la potencia de funcionamiento, en su funcionamiento tiene una relación entre la diferencia de temperatura entre sus placas y la potencia suministrada, este prototipo utiliza una célula TEC1-12706, soporta un máximo de 105 W, con una diferencia de temperatura entre sus placas de 75 ºC, el uso dado en este proyecto utiliza menos del 50 % de su capacidad total.

Figura 30. **Placa Peltier TEC1-12706**

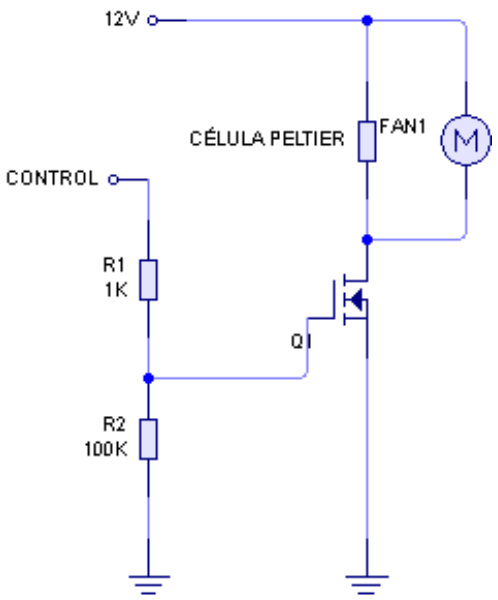


Fuente: DH Gate. *Enfriador termoeléctrico Peltier.* https://es.dhgate.com/product/1-piece-tec1-12706-40x40mm-tec-thermoelectric/421645374.html. Consulta: 19 de enero de 2020.

3.2.1.3. Circuito de conmutación

La administración de la potencia de la célula Peltier y el ventilador se maneja por medio de un MOSFET canal N conectado en configuración fuente común, este dispositivo cuenta con baja disipación de calor y muy buena respuesta al usarse como interruptor.

Figura 31. **Diagrama del circuito de conmutación**



Fuente: elaboración propia, empleando Livewire 2004.

3.2.2. Actuador de sombra

Para reducir la luminiscencia incidente este prototipo cuenta con un dispositivo que atenúa la luz, se construyó en un marco rectangular con dos rodillos en los extremos lejanos, cuenta con tres diferentes niveles de atenuación, está formado por tres segmentos de nylon de diferente espesor, al ser un material traslucido, permite que pase un porcentaje del total incidente.

Figura 32. **Rollo de nylon del actuador de sombra**

Fuente: elaboración propia.

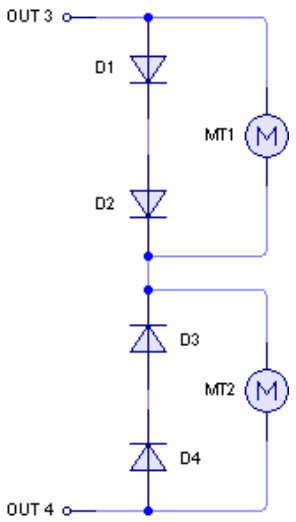
El desplazamiento del nylon se realiza por medio de dos motores DC, uno encargado de aumentar de nivel de atenuación y otro de reducirlo, ambos motores se encuentran conectados a un puente H L298N, utiliza un sensor de paso que indica cuando se ha realizado un cambio de nivel por completo, de esta forma desactiva los motores.

Figura 33. **Sensor de desplazamiento de nivel de sombra**

Fuente: elaboración propia.

El mecanismo del actuador de sombra requiere de trabajo del motor del movimiento activo y una contribución mínima del motor que corresponde al desplazamiento inactivo, para ello ambos motores están conectados en serie entre ellos y a un arreglo de diodos que limita el voltaje en el motor que realiza contribución mínima en cada momento.

Figura 34. **Diagrama de conexión de motores del actuador de sombra**



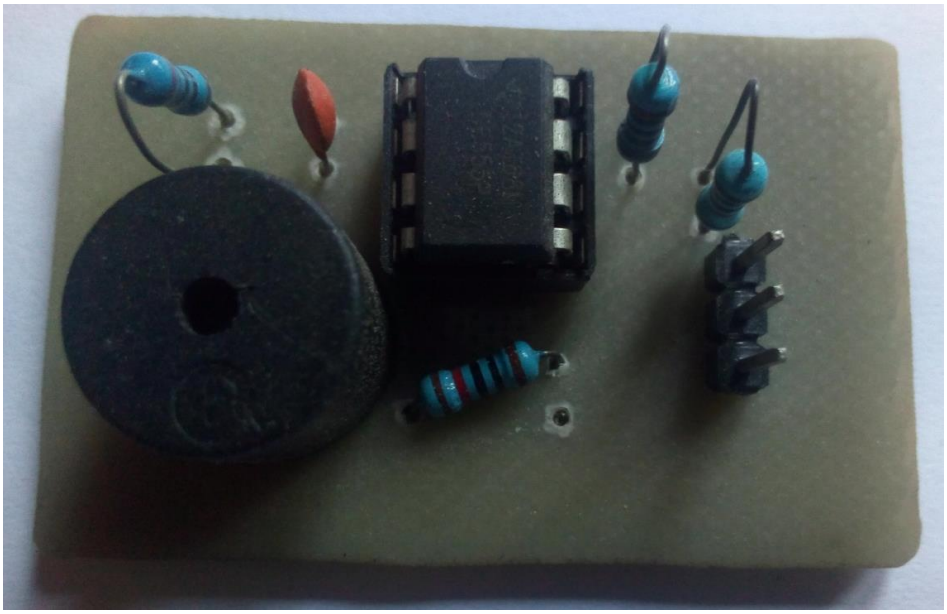
Fuente: elaboración propia, empleando Livewire 2004.

3.2.3. Actuador de alertas

Se ha incorporado un circuito que indica de forma sonora la necesidad de intervención del usuario. Utiliza un circuito integrado NE555 en configuración multivibrador astable, para ser audible genera una señal cuadrada a una frecuencia de 500 Hz con un ciclo de trabajo del 50 %.

Su activación o desactivación es controlada desde la Raspberry Pi 3, con un terminal conectado al terminal de RESET del NE555, las características del funcionamiento del circuito son independientes de la Raspberry Pi 3, si en futuros prototipos se considera necesario otro dispositivo para alertar al usuario es posible agregarlo simplemente conectando esta terminal de control.

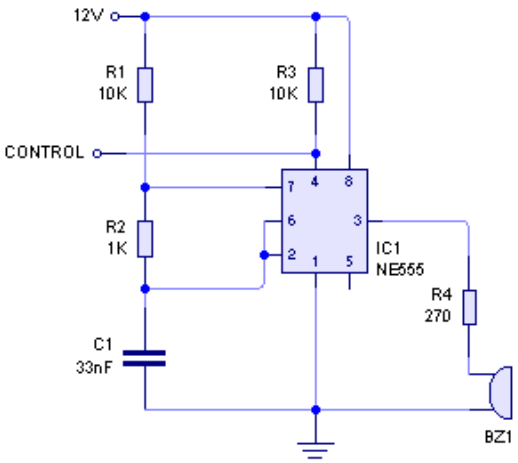
Figura 35. **PCB del actuador de alertas**



Fuente: elaboración propia.

El dispositivo que convierte la señal generada en sonido es un transductor buzzer incluido en la misma PCB.

Figura 36. **Diagrama del actuador de alertas**

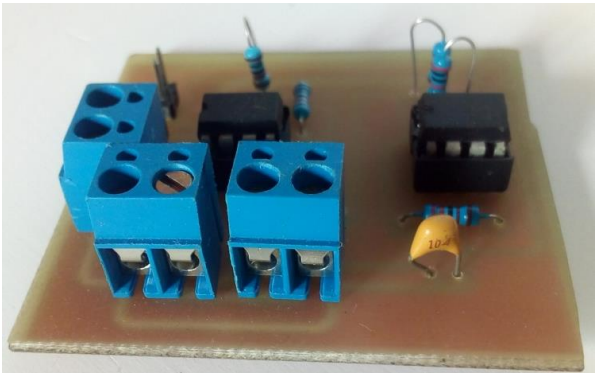


Fuente: elaboración propia, empleando Livewire 2004.

3.2.4. Seguidor solar

Considerando que durante un día normal el ángulo de incidencia de luz solar cambia, el actuador de sombra está colocado en un marco rectangular móvil, el movimiento del marco depende de un circuito seguidor solar también conocido como girasol, dicho circuito está formado por un amplificador operacional dual con ambos amplificadores en configuración de comparador, utiliza como entradas las terminales de dos LDR, ambas en los extremos más lejanos del marco.

Figura 37. **PCB del seguidor solar**

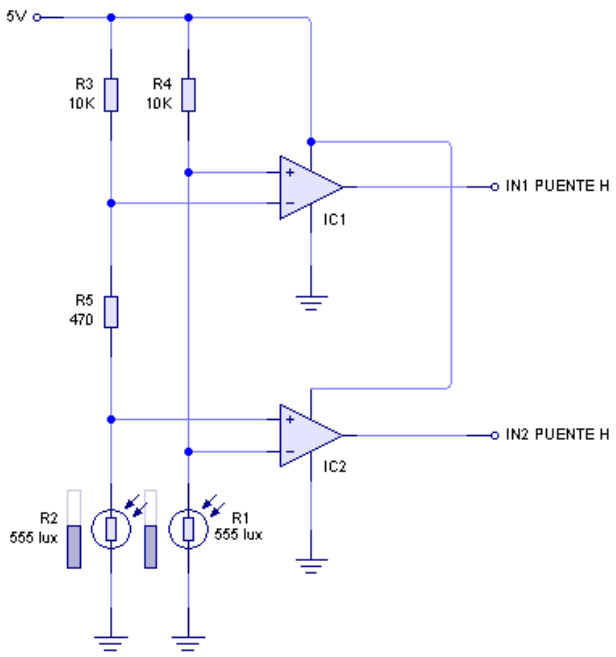


Fuente: elaboración propia.

Si la luz incidente en un LDR cambia respecto al otro, su resistencia cambia, dando lugar a un cambio de voltaje en las entradas del comparador, lo que dispara una señal alta en una salida de un comparador, dicha señal va hacia un puente H que controla el motor a cargo del movimiento del marco, el motor gira en la dirección que el circuito considera compensara la diferencia de luz incidente en los LDR.

El funcionamiento de este actuador es totalmente independiente del sistema, en cuanto se suministra alimentación al prototipo el actuador se pondrá en marcha.

Figura 38. **Diagrama del seguidor solar**



Fuente: elaboración propia, empleando Livewire 2004.

3.3. Circuito de respaldo

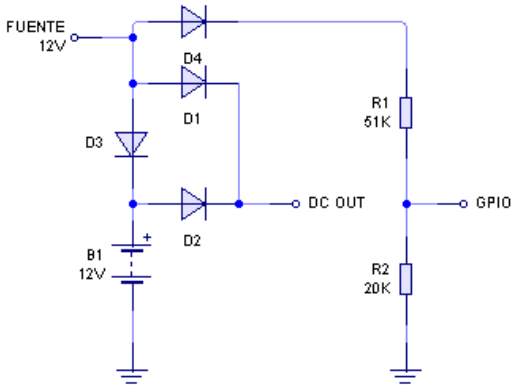
Al prescindir del suministro eléctrico el prototipo cuenta con un respaldo energético, permite disponer del tiempo suficiente para cerrar el programa y apagarse de forma adecuada. El circuito se encarga de indicar la ausencia o presencia de energía eléctrica.

Figura 39. **PCB del circuito de respaldo**

Fuente: elaboración propia.

El respaldo consiste en una compuerta OR elaborada con diodos Schottky la alimentación es posible desde la fuente o bien desde la batería de respaldo, también cuenta con un arreglo de diodos comunes que cargan la batería de respaldo y proveen de corriente a un divisor de voltaje que da la señal de presencia de suministro eléctrico, el divisor de voltaje reduce el voltaje de señal a 3,3 V protegiendo los GPIO de la Raspberry.

Figura 40. **Diagrama del circuito de respaldo**



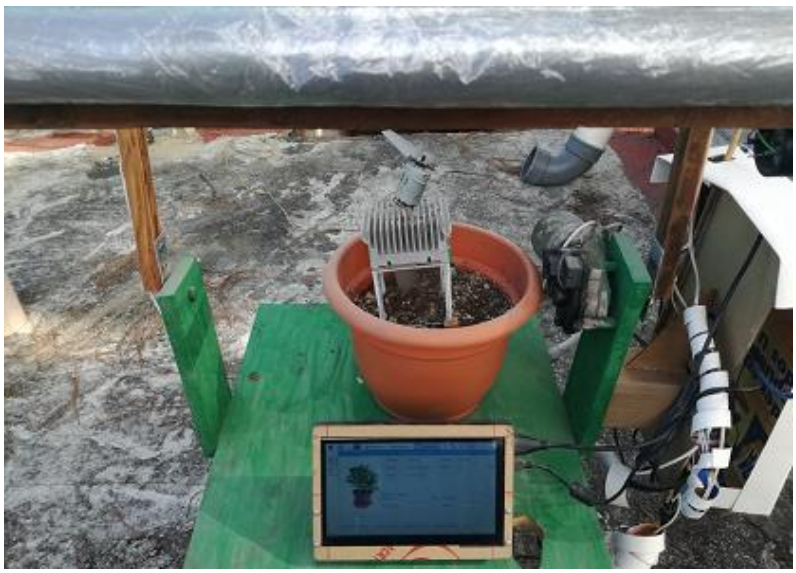
Fuente: elaboración propia, empleando Livewire 2004.

4. PUESTA EN MARCHA DEL PROTOTIPO

4.1. Prueba del prototipo al aire libre

En la figura 40 se muestra la vista frontal del prototipo completo con la interfaz de usuario abierta justo antes de iniciar el sistema, la prueba se realizó con una maceta mostrada en la figura, dentro de esta se encuentra el actuador de condensación distinguible por su unidad de refrigeración artificial, a la derecha de la maceta se observa el actuador del seguidor solar y finalmente a la derecha de este se halla el gabinete que contiene el minicomputador, microcontrolador, módulo step-down, PCB del seguidor solar, del actuador de alertas y del circuito de respaldo.

Figura 41. **Sistema listo para iniciar**

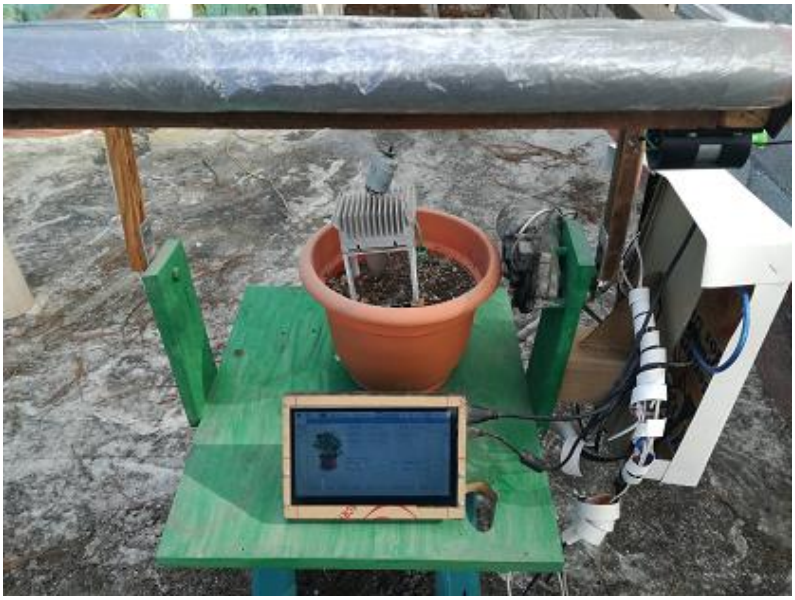


Fuente: elaboración propia.

Figura 42. **Interfaz con sistema completo iniciado**

Fuente: elaboración propia.

Figura 43. **Sistema completo iniciado**



Fuente: elaboración propia.

Figura 44. **Actuador de sombra en funcionamiento**

Fuente: elaboración propia.

Figura 45. **Actuador de condensación en funcionamiento**

Fuente: elaboración propia.

Figura 46. **Actuador de condensación empañado**

Fuente: elaboración propia.

Figura 47. **Actuador del seguidor solar en funcionamiento**

Fuente: elaboración propia.

Figura 48. **Vista completa del sistema**

Fuente: elaboración propia.

Figura 49. **Vista completa del sistema lateral**

Fuente: elaboración propia.

4.2. Consideraciones para aplicación escalable del proyecto

Conservando los métodos tradicionales de riego, la implementación de este sistema tiene un impacto en la optimización del uso de recursos, esto significa controlar el encendido y caudal de riego de una motobomba, colocar una estación de telemetría en el área de interés e incorporar un sistema de suministro energético para aire libre.

Tabla VI. **Costo por estación de telemetría con riego tradicional**

Artículo	Costo unitario	Unidades	Costo total
Panel solar de 20w	Q 200,00	1	Q 200,00
Batería acido plomo 12V 4Ah	Q 170,00	1	Q 170,00
Regulador de 12V a 5V	Q 100,00	1	Q 100,00
Controlador de carga de batería	Q 360,00	1	Q 360,00
Caja para intemperie	Q 220,00	1	Q 220,00
Tarjeta GSM con SIM	Q 300,00	1	Q 300,00
Sensor 200ss	Q 500,00	1	Q 500,00
Sensor DS18B20	Q 50,00	1	Q 50,00
Sensor BH1750	Q 35,00	1	Q 35,00
Tarjeta de conexiones	Q 65,00	1	Q 65,00
Módulo de control de RPM	Q 2 000,00	1	Q 2000,00
Microcontrolador Arduino Uno	Q 95,00	1	Q 95,00
Instalación	Q 1 000,00	1	Q 1 000,00
TOTAL			Q 5 095,00

Fuente: elaboración propia.

Cada estación de telemetría abarca el área que corresponde a la capacidad de la motobomba instalada que suele ser de 4 a 5 hectáreas, dicha área puede ampliarse incorporando válvulas solenoide con un costo de Q 15 000,00 cada una.

Tabla VII. **Costo por central de monitoreo con riego tradicional**

Artículo	Costo unitario	Unidades	Costo total
Batería acido plomo 12V 4Ah	Q 170,00	1	Q 170,00
Regulador de 12V a 5V	Q 100,00	1	Q 100,00
Cargador de batería	Q 200,00	1	Q 200,00
Tarjeta GSM con SIM	Q 300,00	1	Q 300,00
Raspberry Pi3b+	Q 640,00	1	Q 640,00
Monitor LCD	Q 500,00	1	Q 500,00
Teclado alfanumérico	Q 250,00	1	Q 250,00
Ratón alámbrico	Q 150,00	1	Q 150,00
Instalación	Q 1 000,00	1	Q 10000,00
TOTAL			Q 12310,00

Fuente: elaboración propia.

El sistema requiere de una sola central de monitoreo, que es capaz de administrar la totalidad de estaciones de telemetría instaladas.

Posteriormente el costo mensual del sistema es de Q 200,00 por cada estación y Q 200,00 de la central, debido al pago de un servicio de telefonía para la tarjeta GSM de cada punto.

Así mismo la mitad de los operadores que se requieren por cada motobomba pueden ser reubicados en otras áreas, ya que únicamente se realizaría inspección del funcionamiento de las estaciones.

CONCLUSIONES

1. Los resultados indican que las características de los sensores utilizados cumplen con los requisitos de precisión y exactitud necesarios para este prototipo, estas se encuentran en la sección 3 de este documento, además con base en lo expuesto, se deduce que la medición de tensión de agua en el suelo proporciona más información que la medida de humedad relativa, lo que hace a los sensores utilizados apropiados para el proyecto por su funcionamiento.

2. La interfaz de usuario desarrollada ofrece las opciones de interés de la planta y del sistema de forma clara, incluye las magnitudes, así como las unidades de las medidas presentes que refresca continuamente.

3. De igual forma el registro de datos contiene toda la información de interés a nivel de usuario y a nivel técnico, al realizar un diagnóstico o estudio del sistema completo este archivo permite comprender de mejor forma el funcionamiento del prototipo, la estructura del código y observar las tendencias del comportamiento del cultivo.

RECOMENDACIONES

1. Colocar el sensor de temperatura y el de tensión de agua del suelo a la profundidad más apropiada para el cultivo, además, encapsular el sensor de luminiscencia en un recipiente hermético y transparente que deberá ser limpiado continuamente.

2. Cerrar la interfaz de usuario cuando se realicen modificaciones al programa o se desee ejecutar de nuevo desde el editor.

3. Realizar revisiones del registro de datos cuando el sistema está detenido o hacer una copia del mismo, para evitar un conflicto entre aplicaciones que tratan de abrir un mismo archivo.

BIBLIOGRAFÍA

1. CHEN LÓPEZ, José. *La influencia de la luz en el crecimiento del cultivo*. [en línea]. <https://www.pthorticulture.com/es/centro-de-formación/la-influencia-de-la-luz-en-el-crecimiento-del-cultivo/>. [Consulta: 18 de abril de 2019].

2. CSSA. *¿Tienen las plantas diferentes metabolismos?* [en línea]. <https://www.infoagro.com/noticias/2019/_tienen_las_plantas_diferentes_metabolismos_.asp>. [Consulta: 19 de abril de 2019].

3. DIAZ, Wilfredo. *Sistema de control.* [en línea]. <https://es.slideshare.net/wilfredodiaz2/sistemas-de-control-50453873>. [Consulta: 20 de abril de 2019].

4. Electrónica Unicrom. *Polarización de los FET.* [en línea]. <https://unicrom.com/polarizacion-del-fet/>. [Consulta: 19 de abril de 2019].

5. Facultad de Agronomía. *Agua en el suelo.* [en línea]. <http://www.fagro.edu.uy/~hidrologia/paisajismo/AGUA%20EN%20EL%20SUELO.pdf>. [Consulta: 19 de abril de 2019].

6. GÓMEZ ESTEBAN, Pedro. *¿Qué es el efecto Peltier?* [en línea]. <https://eltamiz.com/2007/08/30/%C2%BFque-es-el-efecto-peltier/>. [Consulta: 10 de julio de 2019].

7. PACHÉS GINER, María. *El agua en el suelo: fuerzas de retención.* [en línea]. <https://riunet.upv.es/bitstream/handle/10251/121154/Pach%C3%A9s%20-%20El%20agua%20en%20el%20suelo.%20Fuerzas%20de%20retenci%C3%B3n.pdf?sequence=1&isAllowed=y>. [Consulta: 19 de abril de 2019].

8. PAPIEWSKI, John. *El circuito integrado 555 y su funcionamiento.* [en línea]. <https://www.puromotores.com/13079969/el-circuito-integrado-555-y-su-funcionamiento>. [Consulta: 19 de abril de 2019].

9. PROGRAMO ERGO SUM. *¿Qué es Raspberry Pi?* [en línea]. <https://www.programoergosum.com/cursos-online/raspberry-pi/232-curso-de-introduccion-a-raspberry-pi/que-es-raspberry-pi>. [Consulta: 12 de octubre de 2019].

APÉNDICES

Apéndice 1. **Programa del prototipo**

```
#Programa del prototipo del sistema de control de luminiscencia y riego
#Elaborado por Samuel Choc
#2020

import os, sys, serial
from time import localtime, strftime, sleep
try:
   # Windows.
   from os import startfile
except ImportError:
   # Otras plataformas.
   from webbrowser import open as startfile
from PyQt4.QtCore import SIGNAL, QEvent
from PyQt4.QtGui import (QApplication, QHBoxLayout, QIcon, QMainWindow,
                         QLabel, QLineEdit, QPushButton, QTreeWidget,
                         QTreeWidgetItem, QVBoxLayout, QWidget, QPixmap,
                         QMessageBox, QDialog, QPaintEvent)

import RPi.GPIO as GPIO
import threading

global elige
elige = []

class confirmar(QDialog):#ventana emergente para confirmar de acciones
   def __init__(self, mensaje):
      super(confirmar, self).__init__(None)
      self.resize(340,140)
      self.setWindowTitle('Confirmar')
```

Continuación del apéndice 1.

```
            self.pregunta = QLabel(mensaje,self)
            self.pregunta.move(40,50)

            self.aceptari = QPushButton('Aceptar', self)
            self.aceptari.setGeometry(60, 85, 80, 30)

            self.cancelari = QPushButton('Cancelar', self)
            self.cancelari.setGeometry(200, 85, 80, 30)

            self.connect(self.aceptari, SIGNAL('clicked()'), lambda: self.hacer(True))
            self.connect(self.cancelari, SIGNAL('clicked()'), lambda: self.hacer(False))

        def hacer(self, d):
            elige.append(d)
            self.close()

    class ventana(QWidget):#ventana general de la interfaz de usuario

        def __init__(self):
            self.ingresar = QApplication(sys.argv)
            super(ventana, self).__init__(None)
            self.flag = 0
            self.flag1 = 0
            self.todo = 'datos.txt'
            self.planta = ''
            self.ingreso = QWidget()
            self.ingreso.resize(740, 410)
            self.ingreso.setWindowTitle('Cuidando de Tus Plantas')
            self.puerto = '/dev/serial/by-id/usb-
Arduino__www.arduino.cc__0043_85734323230351407291-if00'
            self.baudrate = 57600

        def inicio(self, u = 0, c = 0):#partes de la interfaz de usuario

            self.imaje = QLabel(self.ingreso)
```

Continuación del apéndice 1.

```
self.imaje.setGeometry(10,10,183,275)
self.imaje.setPixmap(QPixmap("logo.jpeg"))

self.n = QLabel('Nombre',self.ingreso)
self.n.move(220,10)

self.p = QLabel('Planta',self.ingreso)
self.p.move(480,10)

self.f = QLabel('Familia',self.ingreso)
self.f.move(220,50)

self.a = QLabel('Edad',self.ingreso)
self.a.move(480,50)

self.a = QLabel('Dias',self.ingreso)
self.a.move(705,50)

self.e = QLabel('Etapa',self.ingreso)
self.e.move(220,90)

self.t = QLabel('Tension de agua\n en el suelo',self.ingreso)
self.t.move(200,188)

self.ut = QLabel('kPa',self.ingreso)
self.ut.move(435,210)

self.l = QLabel('Lumenes',self.ingreso)
self.l.move(490,210)

self.ul = QLabel('lx',self.ingreso)
self.ul.move(705,210)

self.c = QLabel('Temperatura',self.ingreso)
self.c.move(200,250)
```

Continuación del apéndice 1.

```
self.uc = QLabel('\N{DEGREE SIGN}C',self.ingreso)
self.uc.move(435,250)

self.i = QLabel('Alerta',self.ingreso)
self.i.move(490,250)

self.w1 = QLabel('Campo "Nombre" Vacio', self.ingreso)
self.w1.move(220,30)
self.w1.hide()

self.w2 = QLabel('Campo "Planta" Vacio', self.ingreso)
self.w2.move(480,30)
self.w2.hide()

self.w3 = QLabel('Campo "Familia" Vacio', self.ingreso)
self.w3.move(220,70)
self.w3.hide()

self.w4 = QLabel('Campo "Edad" Vacio', self.ingreso)
self.w4.move(480,70)
self.w4.hide()

self.w5 = QLabel('Campo "Etapa" Vacio', self.ingreso)
self.w5.move(220,110)
self.w5.hide()

self.w6 = QLabel('No se encuentra ninguna planta con esos datos', self.ingreso)
self.w6.move(220,150)
self.w6.hide()

self.w7 = QLabel('Conexion Fallida', self.ingreso)
self.w7.move(220,290)
self.w7.hide()

self.Nombre = QLineEdit(self.ingreso)
```

Continuación del apéndice 1.

```
self.Nombre.setGeometry(320, 10, 140, 20)

self.Planta = QLineEdit(self.ingreso)
self.Planta.setGeometry(560, 10, 140, 20)

self.Familia = QLineEdit(self.ingreso)
self.Familia.setGeometry(320, 50, 140, 20)

self.Edad = QLineEdit(self.ingreso)
self.Edad.setGeometry(560, 50, 140, 20)

self.Etapa = QLineEdit(self.ingreso)
self.Etapa.setGeometry(320, 90, 140, 20)

self.Tension = QLineEdit(self.ingreso)
self.Tension.setGeometry(290, 210, 140, 20)

self.Lumenes = QLineEdit(self.ingreso)
self.Lumenes.setGeometry(560, 210, 140, 20)

self.Temperatura = QLineEdit(self.ingreso)
self.Temperatura.setGeometry(290, 250, 140, 20)

self.Alerta = QLineEdit(self.ingreso)
self.Alerta.setGeometry(560, 250, 140, 20)

self.agrega = QPushButton('Agregar Planta', self.ingreso)
self.agrega.setGeometry(20, 350, 120, 30)

self.guarda = QPushButton('Guardar Planta', self.ingreso)
self.guarda.setGeometry(160, 350, 120, 30)

self.busca = QPushButton('Buscar Planta', self.ingreso)
self.busca.setGeometry(300, 350, 120, 30)
```

Continuación del apéndice 1.

```
            self.elimina = QPushButton('Eliminar Planta', self.ingreso)
            self.elimina.setGeometry(440, 350, 120, 30)

            self.inicia = QPushButton('Iniciar Sistema', self.ingreso)
            self.inicia.setGeometry(580, 350, 120, 30)

            self.detiene = QPushButton('Detener Sistema', self.ingreso)
            self.detiene.setGeometry(580, 350, 120, 30)
            self.detiene.hide()

            self.ingreso.connect(self.agrega, SIGNAL('clicked()'), lambda: self.agregar())
            self.ingreso.connect(self.guarda, SIGNAL('clicked()'), lambda: self.guardar(self.Nombre,
self.Planta, self.Familia, self.Edad, self.Etapa))
            self.ingreso.connect(self.busca, SIGNAL('clicked()'), lambda: self.buscar(self.Nombre,
self.Planta, self.Familia, self.Edad, self.Etapa))
            self.ingreso.connect(self.elimina, SIGNAL('clicked()'), lambda:
self.eliminar(self.Nombre, self.Planta, self.Familia, self.Edad, self.Etapa))
            self.ingreso.connect(self.inicia, SIGNAL('clicked()'), lambda: self.iniciar(self.Nombre,
self.Planta, self.Familia, self.Edad, self.Etapa))
            self.ingreso.connect(self.detiene, SIGNAL('clicked()'), lambda: self.detener())

            self.ingreso.show()

            sys.exit(self.ingresar.exec_())

        def vaciarcampos(self):#vaciado de campos de la interfaz
            self.Nombre.setText('')
            self.Planta.setText('')
            self.Familia.setText('')
            self.Edad.setText('')
            self.Etapa.setText('')
            self.Tension.setText('')
            self.Lumenes.setText('')
            self.Temperatura.setText('')
            self.Alerta.setText('')
```

Continuación del apéndice 1.

```
def ocultaradvertencias(self):#oculta las advertencias de la interfaz
    self.w1.hide()
    self.w2.hide()
    self.w3.hide()
    self.w4.hide()
    self.w5.hide()
    self.w6.hide()
    self.w7.hide()

def agregar(self):#reinicia la interfaz
    self.vaciarcampos()
    self.ocultaradvertencias()

def guardar(self, n, p, f, a, e):#agrega una nueva planta al archivo
    self.flag = 0
    self.flag1 = 0
    self.revisar(n,p,f,a,e)
    t = localtime()
    a = str(t[0])+','+str(t[1])+','+str(int(t[2])-int(a.text()))
    if(self.flag == 1):
        general = open(self.todo,'a')
        general.write(n.text()+','+p.text()+','+f.text()+','+a+','+e.text()+'\n')
        general.close()

def buscar(self, n, p, f, a, e):#revisa la existencia de una planta en el archivo
    self.ocultaradvertencias()
    general = open(self.todo, 'r')
    for i in general:
        j = i.split(',')
        if(j[0] == n.text() or j[1] == p.text() or j[2] == f.text()):
            self.flag1 = 1
            self.Nombre.setText(j[0])
            self.Planta.setText(j[1])
            self.Familia.setText(j[2])
            t = localtime()
```

Continuación del apéndice 1.

```
            e = 365*(t[0]-int(j[3])) + 30*(t[1]-int(j[4])) + t[2]-int(j[5])
            self.Edad.setText(str(e))
            self.Etapa.setText(j[6])
            self.w6.hide()
            break
        else:
            self.w6.show()
    general.close()

def eliminar(self, n, p, f, a, e):#borra la planta del archivo
    e = confirmar('Seguro que desea eliminar la planta?')
    e.exec_()
    if elige[0]:
        self.vaciarcampos()
        print('si')
        elige.clear()

def iniciar(self, n, p, f, a, e):#inicia el sistema cuando se confirma la orden
    self.revisar(n,p,f,a,e)
    if(self.flag == 1):
        e = confirmar('Seguro que desea iniciar el sistema?')
        e.exec_()
        if (elige[0]):
            self.inicia.hide()
            self.detiene.show()
            self.medir()
            self.administrar()
            self.mostrar()
            elige.clear()

    elif(self.flag == 0):
        elige.clear()

def detener(self):#detiene el sistema cuando se confirma la orden
    e = confirmar('Seguro que desea detener el sistema?')
```

Continuación del apéndice 1.

```
        e.exec_()

        if elige[0]:
            self.detiene.hide()
            self.inicia.show()
            self.ocultaradvertencias()
            elige.clear()
            stop_threads = True
            GPIO.cleanup()

    def medir(self):#realiza las mediciones del sistema
        clase = recibirdatos()
        hilo2 = threading.Thread(target = clase.recibir, name='Medidas',daemon = False)
        hilo2.start()

    def mostrar(self):#actualiza los datos en la interfaz del usuario
        while 1:
            QApplication.processEvents()

            archivo = open('ultimoresultado.txt', 'r')
            for i in archivo:
                self.Tension.setText(str(int(float(i.split(',')[0]))))
                self.Lumenes.setText(i.split(',')[1])
                self.Temperatura.setText(i.split(',')[2])
                if(i.split(',')[3] == '3'):
                    self.Alerta.setText('Falla de Suministro')
                elif(i.split(',')[3] == '2'):
                    self.Alerta.setText('Error DS18B20')
                elif(i.split(',')[3] == '1'):
                    self.Alerta.setText('Error Tension')
                elif(i.split(',')[3] == '0'):
                    self.Alerta.setText('Sin Alertas')
            archivo.close()
            self.registro()
            sleep(3)
```

Continuación del apéndice 1.

```
def registro(self):#guarda las medidas y nombre de la planta
    archivo = open('comp.txt', 'r+')
    for i in archivo:
        j = i+'\n'
    archivo.close()
    archivo = open(self.Nombre.text()+'.txt','a+')
    archivo.write(self.Nombre.text()+','+self.Planta.text()+','+self.Familia.text()+',')
    archivo.write(self.Edad.text()+','+self.Etapa.text()+','+self.Tension.text()+',')
    archivo.write(self.Lumenes.text()+','+self.Temperatura.text()+','+self.Alerta.text()+','+j)
    archivo.close()

def administrar(self):#administra los actuadores
    c = control()
    hilo1 = threading.Thread(target = c.proceso, name='Actuadores')
    hilo1.start()

def revisar(self, n, p, f, a, e):#verifica microcontrolador y datos de los campos

    #asignar texto en campos editables a variables locales
    nombre = n.text()
    planta = p.text()
    familia = f.text()
    edad = a.text()
    etapa = e.text()

    #intenta conectar con el microcontrolador y advierte si falla
    try:
        arduino = serial.Serial(self.puerto, self.baudrate)
        self.flag = 1
    except:
        try:
            arduino.close()

        except:
            print ('Puerto no Conectado')
```

Continuación del apéndice 1.

```
            self.w7.show()
            self.flag = 0
            print ('Error al Conectar')

        #revisa que exista texto en todos los campos editables
        if(nombre != '' and planta != '' and familia != '' and edad != '' and etapa != ''):
            self.flag = 1*self.flag
        else:
            self.flag = 0*self.flag

        #mostrar u ocultar los avisos para cada campo vacio
        if(nombre == ''):
            self.w1.show()
        else:
            self.w1.hide()

        if(planta == ''):
            self.w2.show()
        else:
            self.w2.hide()

        if(familia == ''):
            self.w3.show()
        else:
            self.w3.hide()

        if(edad == ''):
            self.w4.show()
        else:
            self.w4.hide()

        if(etapa == ''):
            self.w5.show()
        else:
            self.w5.hide()
```

Continuación del apéndice 1.

```
class procesardatos(object):#procesa las medidas obtenidas de los sensores
    def __init__(self):
        self.aviso = 0
        self.vref = 5.00
        self.voltaje1_s = 0.00
        self.voltaje2_s = 0.00
        self.voltaje_p = 0.00
        self.niveles = 1023
        self.resistencia = 7439
        self.resistencia_s = 0.00
        self.temperatura = 0.00
        self.temperatura_d = -127.0
        self.tension = 0.00
        self.lumenes = 0.00
        self.medida = 'ultimamedida.txt'
        self.resultado = 'ultimoresultado.txt'

    def lecturaanalogicaavoltaje(self):#procesa las medidas del convertidor ADC
        archivo = open(self.medida, 'r')
        for i in archivo:
            j = i.split(',')
            la1 = int(j[2])
            la2 = int(j[3])
            self.lumenes = float(j[1])
            self.temperatura = float(j[0])

        archivo.close()

        if(la1 == 1023 and la2 == 0):
            self.aviso = 1
        else:
            self.voltaje1_s = (la1*self.vref)/self.niveles
            self.voltaje2_s = self.vref*(1 - (la2/self.niveles))
            self.voltaje_p = (self.voltaje1_s + self.voltaje2_s)/2
            self.aviso = 0
```

Continuación del apéndice 1.

```
                if(self.temperatura == self.temperatura_d):
                    self.aviso = 2

                self.voltajearesistencia()

        def voltajearesistencia(self):#procesa las medidas de voltaje
            self.resistencia_s = (self.resistencia*self.voltaje_p)/(self.vref - self.voltaje_p)
            self.resistenciaatension()

        def resistenciaatension(self):#procesa las medidas de resistencia
            if(self.resistencia_s < 550):
                self.tension = 0.00
            elif(self.resistencia_s < 1000):
                self.tension  =  -20.00*((self.resistencia_s/1000.00)*(1.00+0.018*(self.temperatura-
24.00))-0.55)
            elif(self.resistencia_s < 8000):
                self.tension                =                (-3.213*(self.resistencia_s/1000.00)-4.093)/(1-
0.009733*(self.resistencia_s/1000.00)-0.01215*self.temperatura)
            elif(self.resistencia_s < 27950 ):
                self.tension                =                -2.246-5.239*(self.resistencia_s/1000.00)*(1-
0.018*(self.temperatura-24.00))-0.06756*((self.resistencia_s/1000.00)**2)*((1-
0.018*(self.temperatura-24.00))**2)
            elif(self.resistencia_s >= 27950):
                self.tension = -200

            archivo = open(self.resultado, 'a')
            archivo.write(str(self.tension)+',')
            archivo.write(str(self.lumenes)+',')
            archivo.write(str(self.temperatura)+',')
            archivo.write(str(self.aviso)+',\n')
            archivo.close()

    class recibirdatos(object):#conecta con el microcontrolador y guarda los datos
```

Continuación del apéndice 1.

```
        def __init__(self):
            self.puerto = '/dev/serial/by-id/usb-
Arduino__www.arduino.cc__0043_85734323230351407291-if00'
            self.baudrate = 57600
            self.listadecaracteres = []
            self.bandera = 0
            self.contador = 0
            self.voltaje = 0
            self.medida = 'ultimamedida.txt'

        def recibir(self):#sincroniza los datos que obtiene del microcontrolador
            while 1:
                try:
                    arduino = serial.Serial(self.puerto, self.baudrate)
                    self.bandera = 0
                    self.contador = 0
                    self.listadecaracteres = []

                    while(self.contador < 4):

                        dato = arduino.readline()
                        dato=str(dato)[2:len(dato)]

                        if (dato == 'inicio' and self.bandera == 0):
                            self.bandera = 1
                        elif(self.bandera == 1):
                            self.listadecaracteres.append(dato)
                            self.contador+=1

                    arduino.close()

                    archivo = open(self.medida, 'w')
                    for i in self.listadecaracteres:
                        #print(i)
                        archivo.write(i)
```

Continuación del apéndice 1.

```
                archivo.write(',')
            archivo.write('\n')
            archivo.close()

        except:
            try:
                arduino.close()

            except:
                print ('Puerto no Conectado')

            print ('Error al Conectar')

        clase = procesardatos()
        clase.lecturaanalogicaavoltaje()

class control(object):#constantes de referencia, el algoritmo y condicionales de control
    def __init__(self):
        self.resultado = 'ultimoresultado.txt'
        self.variables = 'valorsalidas.txt'
        self.pwm_condensador = 0
        self.aumentar_sombra = 0
        self.reducir_sombra = 0
        self.salida_condensador = 35
        GPIO.setmode(GPIO.BOARD)
        GPIO.setwarnings(False)
        self.n_s = 0
        self.t_p = 0
        self.t_a = -100
        self.t_r = -40
        self.l = 0
        self.l_min = 10000
        self.l_max = 100000
        self.temp = 0
        self.c_p = 5.0
```

Continuación del apéndice 1.

```
        self.c_d = 1.0
        self.c_i = 5.0
        self.c_s = 0
        self.c_m = 0
        self.c_h = 0
        self.bandera = 0
        self.salida_alarma = 32
        self.detector = 38
        self.salida_filtro_1 = 36
        self.salida_filtro_2 = 37
        self.sensor_1 = 38
        self.sensor_2 = 40
        self.lectura_1 = 0
        self.lectura_2 = 0
        GPIO.setup(self.salida_alarma,GPIO.OUT)
        GPIO.setup(self.detector,GPIO.IN, pull_up_down = GPIO.PUD_DOWN)
        GPIO.setup(self.salida_filtro_1,GPIO.OUT)
        GPIO.setup(self.salida_filtro_2,GPIO.OUT)
        GPIO.setup(self.sensor_1,GPIO.IN, pull_up_down = GPIO.PUD_UP)
        GPIO.setup(self.sensor_2,GPIO.IN, pull_up_down = GPIO.PUD_UP)
        GPIO.setup(self.salida_condensador,GPIO.OUT)
        self.pwm = GPIO.PWM(self.salida_condensador, 100)
        self.pwm.start(0)
        self.falla = 0

    def proceso(self):#condicionales y algoritmo de control del sistema
        while 1:
            archivo = open(self.resultado, 'r')
            for i in archivo:
                self.t_p = float(i.split(',')[0])
                self.l = float(i.split(',')[1])
                self.temp = float(i.split(',')[2])
            archivo.close()
```

Continuación del apéndice 1.

```
                self.pwm_condensador = (self.c_p*(self.t_p - self.t_r))*(-1) + self.c_i*self.c_m*(self.t_r
- self.t_p)
                if self.pwm_condensador < 0:
                    self.pwm_condensador = 0
                elif self.pwm_condensador > 100:
                    self.pwm_condensador = 100

                print(str(self.pwm_condensador))
                self.pwm.ChangeDutyCycle(self.pwm_condensador)

                self.t_a = self.t_p

                if (self.c_m == 15 and self.pwm_condensador > 50 and self.n_s < 3 and self.l >
self.l_max):
                    hilo4 = threading.Thread(target = self.moversombra(36,40),
name='Condensador')
                    hilo4.start()
                    GPIO.output(self.salida_alarma, 0)
                    self.c_m = 0
                    self.bandera = 0
                    self.n_s += 1
                elif (self.c_m == 15 and self.pwm_condensador > 50 and self.n_s == 3):
                    self.bandera = 1
                elif (self.c_m == 15 and self.pwm_condensador == 0 and self.n_s > 0):
                    hilo4 = threading.Thread(target = self.moversombra(37,40),
name='Condensador')
                    hilo4.start()
                    GPIO.output(self.salida_alarma, 0)
                    self.c_m = 0
                    self.bandera = 0
                    self.n_s -= 1
                elif (self.c_m == 15 and self.pwm_condensador == 0 and self.n_s == 0):
                    self.bandera = 1

                if self.c_s < 60:
```

Continuación del apéndice 1.

```
            self.c_s += 5
        else:
            self.c_s = 5
            if self.c_m < 60:
                self.c_m += 1
            elif (self.c_m == 60 and self.bandera == 1):
                GPIO.output(self.salida_alarma, 1)
                self.c_m = 1

        self.registro_comp()
        self.falla = GPIO.input(self.detector)
        print(self.falla)
        if self.falla == 0:
            GPIO.output(self.salida_alarma, 1)
            sleep(5)
            os.system('sudo poweroff')
        sleep(5)

    def registro_comp(self):#guarda todos los datos de la planta y del sistema
        archivo = open('comp.txt','w+')
        t = localtime()
        archivo.write(str(self.t_r)+','+str(self.t_p)+','+str(self.t_a)+',')
        archivo.write(str(self.pwm_condensador)+','+str(self.n_s)+','+str(self.bandera)+',')
        archivo.write(str(self.c_m)+','+str(self.falla)+','+str(t[2])+',')
        archivo.write(str(t[1])+','+str(t[0])+','+str(t[3])+':'+str(t[4])+':'+str(t[5]))
        archivo.close()

    def moversombra(self,salida, sensor):#administra el actuador de sombra
        lectura = 0
        while 1:
            GPIO.output(salida, 1)
            if(GPIO.input(sensor) == 0 and lectura == 0):
                lectura = 1
            elif(GPIO.input(sensor) == 1 and lectura == 1):
                lectura = 0
```

Continuación del apéndice 1.

```
                GPIO.output(salida, 0)
                break

def run():
    v = ventana()
    v.inicio()

run()
```

Fuente: elaboración propia, empleando Python 3.5.