



UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

SISTEMA SEMI-AUTOMÁTICO DE ASISTENCIA PARA CULTIVOS EN HUERTOS
URBANOS Y PERIURBANOS

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN

BENJAMÍN ALONSO ZAMORA ZAVANDO

PROFESOR GUÍA:
SERGIO OCHOA DELORENZI

MIEMBROS DE LA COMISIÓN:
DORIS SÁEZ HUEICHAPAN
MARISA ERNST ELIZALDE
JÉRÉMY BARBAY

SANTIAGO DE CHILE
2021

Resumen

La agricultura de precisión ha tomado relevancia en la producción agrícola, principalmente dada la creciente integración de la tecnología en esta área y en nuestras vidas, y también porque la evidencia científica ha demostrado que la implementación de estas estrategias de gestión llevan a una mejora considerable del uso de recursos, la productividad, calidad, rentabilidad y sustentabilidad de los cultivos tecnificados. En consecuencia, este nuevo paradigma amplía el espectro de posibilidades para los profesionales científico-tecnológicos, abriendo oportunidades en la generación de productos o servicios que anteriormente estaban limitados al área industrial y del hogar.

Aprovechando esas oportunidades, este trabajo de memoria contempla el desarrollo de un sistema semi-automático de asistencia al cultivo en huertos urbanos y periurbanos, utilizando tecnologías de bajo costo y Open Source para su desarrollo. Este sistema cuenta con un módulo de control y diversos periféricos. El primero, basado en lógica difusa, es capaz de automatizar procesos clave del cultivo, para lograr la reducción tanto del tiempo invertido por el cultivador, como los conocimientos y experiencia requeridos. Particularmente, el sistema automatiza los procesos de riego y ventilación para el control de las variables del ecosistema, notificando al usuario ante los cambios de estado de estos actuadores. Los periféricos utilizados son principalmente sensores de humedad y temperatura (ambiental y de suelo), un sistema de ventilación/extracción y otro de riego por goteo. Estos periféricos se comunican con el módulo de control utilizando el protocolo de comunicación Mosquitto y diversos componentes electrónicos como intermediarios. Si las variables monitoreadas no alcanzan valores aceptables entonces el sistema envía alertas al cultivador para que éste tome conocimiento de la situación de su cultivo.

Los resultados son presentados por medio de la interfaz gráfica del software Grafana y sus funcionalidades. Las pruebas realizadas para evaluar el sistema, se desarrollan en una carpa de cultivo de interior con dos albahacas tradicionales, además de una serie de dispositivos electrónicos necesarios para completar este escenario. Los criterios de desempeño se basan en varias métricas como la reducción del trabajo invertido en el cuidado, la eficiencia y precisión del control de las variables, la optimización del uso del agua de riego y la flexibilidad entregada al usuario para personalizar la dinámica y parámetros del sistema. Bajo estos aspectos se logran resultados favorables y alentadores, con posibles extensiones como parte del trabajo a futuro.

Agradecimientos

Gracias a mi familia por siempre apoyarme y hacer posible mis estudios fuera de casa, sin ustedes nada de esto habría sido posible. También extendo estos agradecimientos a toda la comunidad Open Source alrededor del mundo que permitieron que este proyecto fuera desarrollado gastando muy poco dinero, y aprendiendo mucho.

Tabla de Contenido

1. Introducción	1
1.1. Desafío abordado	2
1.2. Objetivos de la memoria	2
1.3. Representación general de la solución	3
1.4. Estructura de la memoria	5
2. Antecedentes	6
2.1. Conceptos básicos	6
2.2. Sistemas de monitoreo de cultivos urbanos	7
2.3. Thinger.io	9
2.4. Message Queue Telemetry Transport	10
2.5. Paho MQTT y InfluxDB	12
2.6. Grafana	12
2.7. BotFather	13
2.8. Sistema de Control Difuso y Scikit Fuzzy	13
2.9. Especificaciones técnicas de componentes del sistema	14
2.9.1. Placas de desarrollo	14
2.9.2. Sensores	17
2.9.3. Componentes escogidos para la solución	20
2.9.4. Costo de la solución	21
2.10. Resumen	22
3. Propuesta de Solución	23
3.1. Principales requisitos de la Solución	23
3.2. Arquitectura lógica	24
3.3. Arquitectura del software implementado	25
3.3.1. Contexto general del software implementado	25
3.3.2. Contenedores del software implementado	27
3.3.3. Componentes del módulo de visualización y monitoreo	29
3.4. Modelo de datos del sistema	30
3.5. Discusión	32
4. Sistema de Control Difuso	33
4.1. Introducción a los controladores difusos	33
4.2. Variables consideradas	37
4.3. Arquitectura del controlador	39

4.4. Conjuntos Difusos	41
4.5. Reglas Difusas	45
4.6. Algoritmo de control implementado	47
4.7. Discusión	49
5. Evaluación de la Solución Propuesta	50
5.1. Escenario de Pruebas	50
5.2. Pruebas realizadas	52
5.2.1. Funcionamiento básico del controlador difuso	52
5.2.2. Control a corto plazo de humedad de sustrato y ambiente	57
5.2.3. Control a largo plazo (10 días)	60
5.2.4. Comparación entre el control a largo plazo y el riego manual	61
5.2.5. Alertas y notificaciones	62
6. Conclusiones y trabajo a futuro	64
Bibliografía	66

Índice de Tablas

2.1. Placas de desarrollo para la implementación del cerebro	15
2.2. Placas de desarrollo para la implementación del gateway	16
2.3. Sensores analizados	20
2.4. Especificaciones técnicas de los sensores del sistema	21
2.5. Costo agregado del sistema propuesto	21
2.6. Comparación productos del mercado con el sistema propuesto	22
4.1. Rangos óptimos para cada variable	42
4.2. Rangos de las funciones de pertenencia para cada variable	42
4.3. Rangos de las funciones de pertenencia del error de cada variable	42
4.4. Rangos de las funciones de pertenencia del delta (del error) de cada variable	43
4.5. Reglas difusas de la humedad de sustrato y el riego	46
4.6. Reglas difusas de la temperatura de sustrato y el riego	46
4.7. Reglas difusas de la humedad ambiente y la ventilación	46
4.8. Reglas difusas de la temperatura ambiente y la ventilación	46
5.1. Casos de prueba y resultados de manipulación de variables	53
5.2. Métricas de desempeño para el control a corto plazo	59
5.3. Métricas de desempeño para control a largo plazo	61
5.4. Resultados de comparación con riego manual	61

Índice de Ilustraciones

1.1. Arquitectura física del sistema	3
2.1. Prototipo en plataforma Thingier.io con ventana de 16hrs.	10
2.2. Diagrama de secuencia de componentes de humedad y riego del sustrato . . .	11
3.1. Arquitectura lógica del sistema	24
3.2. Instancia de la arquitectura lógica del sistema propuesto	25
3.3. Diagrama de contexto del sistema	26
3.4. Diagrama de contenedores del módulo de visualización y monitoreo experto .	27
3.5. Diagrama de contenedores del módulo de gestión semi-automática de cultivos	28
3.6. Diagrama de componentes del módulo de visualización y monitoreo	29
3.7. Modelo de datos del sistema desarrollado	30
4.1. Funciones de pertenencia clásicas [29]	34
4.2. Estructura de un modelo difuso tipo Mamdani [29]	35
4.3. Métodos de defusificación [24]	36
4.4. Estructura estándar de un controlador PI difuso [29]	36
4.5. Proceso de cálculo de una acción de control	38
4.6. Controlador de riego (superior) y ventilación (inferior)	40
4.7. Funciones de pertenencia para los conjuntos de error y su incremento (delta)	43
4.8. Funciones de pertenencia para el error y su incremento (humedad de sustrato)	44
4.9. Funciones de pertenencia para el error y su incremento (temperatura de sustrato)	44
4.10. Funciones de pertenencia para el error y su incremento (humedad ambiente)	44
4.11. Funciones de pertenencia para el error y su incremento (temperatura ambiente)	45
4.12. Función de pertenencia de los actuadores (riego y ventilación)	45
4.13. Ejemplo de activación del riego según la humedad de sustrato	47
5.1. Instancia de la arquitectura física	50
5.2. Interfaz gráfica con información de las últimas 24 hrs.	52
5.3. Resultados variación rápida de humedad de sustrato	54
5.4. Resultados variación rápida de temperatura de sustrato	55
5.5. Resultados variación rápida de humedad ambiente	56
5.6. Resultados variación rápida de temperatura ambiente	57
5.7. Resultados de control de Humedad de Sustrato	58
5.8. Resultados de control de Humedad de Ambiente	59
5.9. Resultados de la primera semana de control	60
5.10. Timelapse de 10 días	62

5.11. Extracto de las últimas 5 notificaciones al usuario	63
---	----

Capítulo 1

Introducción

La evolución de los dispositivos electrónicos ha implicado una integración cada vez mayor de estos en nuestra vida. Algunos ejemplos son las tecnologías para controlar diversas partes de una casa (domótica), los robots domésticos, las tecnologías de monitoreo remoto y los sistemas automatizados de carga (urbótica), etcétera. En resumen, el mercado y sus productos tienden a incorporar cada vez más las tecnologías, aplicaciones y dispositivos asociados al concepto de Internet de las Cosas (IoT - Internet of Things) [13, 34].

Este nuevo escenario tecnológico nos hace pensar no sólo en estas soluciones como un símbolo de comodidad y lujo, sino como una herramienta útil para apoyar actividades en diversos dominios de aplicación, como son la agricultura y la ecología. Nuestro ecosistema se basa en la capacidad que tenemos de convivir en armonía con nuestro entorno, tanto social (humano y animal), como cultural, respetando las tradiciones y nuestro vínculo con la naturaleza.

Dado lo anterior, es importante concientizar sobre el impacto que tiene en el planeta nuestra forma de vivir, y sobre todo acerca de las repercusiones de estas en eventos tan devastadores como el cambio climático y las sequías [33]. Esto no se limita solamente a la transformación de nuestros hábitos cotidianos como acostumbramos a enseñar, sino que compete a toda la industria agrícola, ganadera, minera y energética (entre otras), que miden su utilidad únicamente desde el punto de vista económico y productivo, olvidando su impacto social y político, y lo poco sustentable de sus prácticas en el largo plazo.

Cada día son más las personas que cultivan plantas en sus hogares, oficinas y barrios. Las razones van desde el consumo propio, hasta una contribución al medio ambiente y la cultura local. Siempre ha existido y existirá gente dispuesta a aprender y experimentar la cosecha de sus propios alimentos, a cuidar y proteger un espacio verde propio o comunitario, y por sobre todo, a seguir manteniendo viva la tradición y pasión del proceso de plantar.

Es conocido el interés de las personas en tener plantas en el hogar; sin embargo, se contraponen muchas veces con la falta de tiempo, iniciativa y/o conocimientos necesarios para cuidarlas apropiadamente. En la búsqueda de soluciones para abordar este problema, surge este trabajo de memoria que pretende conectar estos dos mundos, a veces tan enfrentados en

nuestra sociedad: naturaleza y tecnología.

El trabajo realizado consiste en el desarrollo de un sistema que asiste al usuario (cultivador), automatizando procesos tediosos pero fundamentales en el cultivo, como son el riego y la ventilación. De esa manera, se busca controlar de forma directa los valores de temperatura y humedad en el ecosistema para mantenerlos dentro de rangos pre-establecidos. El objetivo es lograr una mejora sustancial en el bienestar y producción de las plantas, permitiendo que personas con poca disponibilidad de tiempo y/o experiencia puedan llevar a cabo la actividad de cultivo.

1.1. Desafío abordado

En este trabajo de memoria se desarrolló un sistema IoT de control semi-automático para apoyar el cultivo en huertos urbanos y periurbanos. El sistema debía brindar asistencia de alto nivel para diversos tipos de cultivadores (desde expertos hasta inexpertos), para así potenciar el surgimiento de espacios de cultivo. Esto involucra el diseño y la implementación de los servicios para monitorear y controlar las variables involucradas en el ecosistema, la comunicación entre los dispositivos que forman parte del escenario tecnológico del sistema (sensores y actuadores), el registro automático de la información recolectada por los sensores, y la generación de notificaciones y alertas para el cultivador en base a los datos almacenados.

Esta memoria surge como parte de un proyecto co-fundado por este memorista, y el Sr. Rodrigo Soria. Este último también realiza su memoria [27] en el marco del proyecto, y su trabajo está enfocado en el desarrollo de una aplicación móvil de apoyo al usuario final, que permita a estas personas administrar sus cultivos e interactuar con cultivadores que son miembros de distintas comunidades, según las plantas de su interés. Por todo lo antes dicho, ambos trabajos de memoria son independientes y complementarios.

1.2. Objetivos de la memoria

En este trabajo de memoria se diseñó e implementó un sistema de software que conecta sensores de diversas variables (humedad y temperatura, tanto de sustrato como del ambiente), con los componentes de riego y ventilación, cuya operación se combina para apoyar el proceso de cuidado de las plantas sembradas. Para lograr este objetivo general, se definieron los siguientes objetivos específicos:

1. Diseñar la arquitectura física y lógica de un sistema de control que permita la automatización de gran parte de las actividades del cultivo en huertos urbanos y periurbanos.
2. Desarrollar un conjunto de servicios de monitoreo y control de variables, así como de envío de notificaciones/alertas al cultivador, en base a la comunicación confiable entre los componentes del sistema y el procesamiento de la información recolectada.
3. Diseñar e implementar un controlador difuso, capaz de automatizar gran parte de las decisiones de cuidado de un cultivo, buscando a su vez optimizar el uso de los recursos disponibles.

1.3. Representación general de la solución

El sistema desarrollado está compuesto de un dispositivo físico (hardware) que se instala en el espacio de cultivo, y una aplicación web (software) para el monitoreo y visualización de los datos del sistema. Es posible utilizar este sistema tanto en invernaderos, como en cultivos de interior o exterior, ocupando todos o algunos de los sensores y actuadores indicados.

A continuación se describe la arquitectura del sistema y el tipo de comunicación entre sus componentes (detallados en la Sección 2.9), explicando su función, interacción y tecnologías utilizadas. La Figura 1.1 ilustra esta representación, utilizando líneas punteadas para simbolizar la comunicación inalámbrica, y líneas sólidas para comunicación directa (cableada):

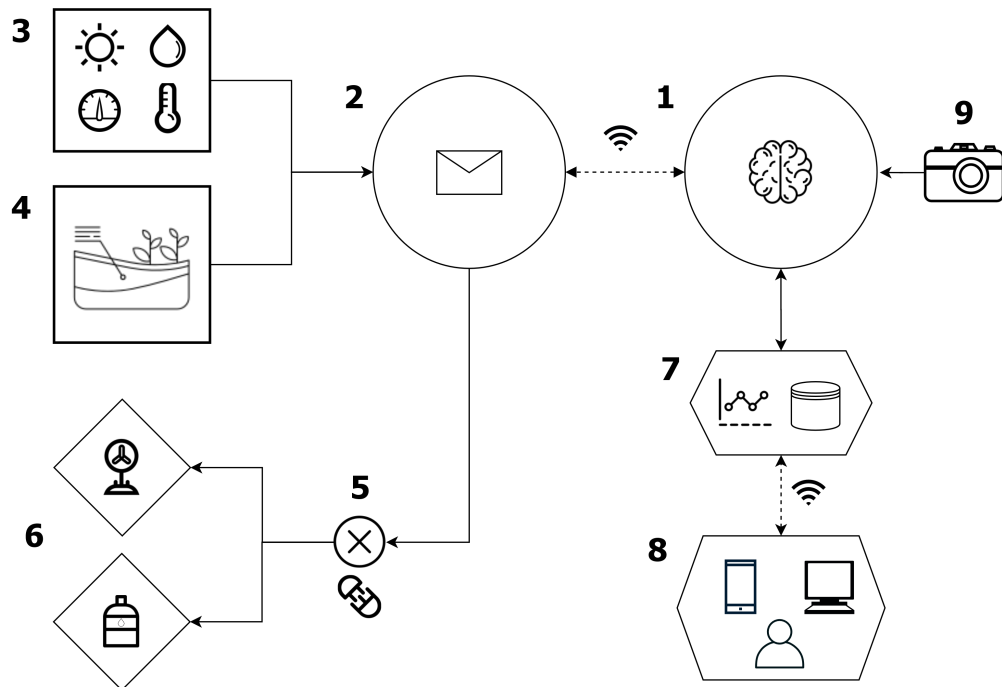


Figura 1.1: Arquitectura física del sistema

1. **Cerebro:** Este componente actúa como servidor del sistema (broker en Mosquitto [19]), esto significa que recibe todos los mensajes publicados en los distintos tópicos (es decir, las mediciones desde los sensores y el estado de los actuadores), y los despacha a sus respectivos suscriptores (componentes que consumen dicha información). Incluye igualmente el software del servidor (7). Además, contiene el código del controlador difuso, que utiliza los datos de los sensores para calcular las acciones automatizadas. Por último, procesa las imágenes de la cámara (9) capturadas en tiempo real para mostrarlas al usuario. Todo lo anterior es implementado en un ordenador de bajo costo Raspberry Pi 3 B+.
2. **Gateway:** Se utiliza como puente entre los sensores (3, 4), actuadores (6) y el cerebro (1). Tiene la misión de recolectar y pre-procesar los datos provenientes de los sensores, para después enviarlos al controlador (cerebro). Además, recibe las señales provenientes de dicho controlador para encender/apagar los interruptores (5) de cada actuador. El código se implementa en una placa NodeMCU v3 ESP8266, que cuenta con pines de lec-

tura/escritura digitales y analógicos, y un módulo WiFi incorporado para comunicarse inalámbricamente utilizando Mosquitto.

3. **Sensor de sustrato:** El conjunto de sensores de este sub-sistema cumple la función de medir variables del sustrato (tierra u otra base para las raíces):
 - (a) **Humedad:** Se mide utilizando un sensor de humedad capacitivo.
 - (b) **Temperatura:** Se mide utilizando un sensor tipo sonda modelo DS18B20.
4. **Sensor de ambiente:** El sensor de este subsistema cumple la función de medir variables ambientales del entorno del cultivo:
 - (a) **Humedad:** Se mide utilizando un sensor BME-280
 - (b) **Temperatura:** Se mide utilizando el mismo sensor
5. **Interruptor:** Para lograr encender o apagar los actuadores (6), necesitamos contar con un interruptor como intermediario. Éste recibe las señales del gateway (2) ya que los actuadores (en general) no tienen forma de comunicarse por defecto con este tipo de dispositivos. En este trabajo de memoria se utilizan dos tipos de interruptores:
 - (a) **Relé 1 Canal 220V:** Este módulo dispone de un transistor para su activación, y con un LED que indica el estado del actuador. Éste permite el tránsito de cargas de 220V a partir de cargas menores ($<5V$), como las manejadas por la placa NodeMCU.
 - (b) **Smart Plug:** Este componente es un tipo de enchufe con módulo Wi-Fi incorporado, que puede ser activado vía web a través de una aplicación móvil o servicios asociados (por ej. IFTTT). En este proyecto se utilizó un modelo de la marca Smart Life, como interfaz al servicio de ventilación.
6. **Actuadores:** Estos equipos modifican el estado del sistema, ya que tienen influencia directa sobre los valores medidos por los sensores:
 - (a) **Riego:** Este componente está compuesto de una bomba peristáltica de 12V, que transporta 40 mL/min de agua desde un estanque plástico de 10L, a través de un sistema de micro-mangueras con terminales de riego por goteo.
 - (b) **Ventilación:** Este componente consiste en un ventilador de 220V Groven KPT-30L y un extractor de aire TT Garden HighPro de 125mm.
7. **Servidor:** Este servicio, incorporado en el cerebro (1), realiza la función de data collector (Paho MQTT) [11], guardando la información de los sensores en una base de datos de serie de tiempo (InfluxDB) [16]. Además, realiza la transformación de estos datos en el software de visualización (Grafana) [12], para generar tableros con gráficas y administrar la configuración de otros servicios (como las notificaciones). El usuario cultivador puede acceder a esta información de forma remota.

8. **Usuario:** Este es el cultivador, el principal usuario al cual apunta este sistema. Es importante mencionar que la información que entrega el sistema también permite que usuarios más calificados puedan desarrollar mejoras al controlador, o evaluar la evolución de sus cultivos de una forma más especializada, dado el nivel de detalle de la misma. Para los usuarios básicos se utiliza la misma estructura, pero la interfaz de usuario que se recomienda es la aplicación desarrollada en la memoria del Sr. Rodrigo Soria [27] (más detalles en la Sección 3.3).
9. **Cámara:** Se utiliza una cámara modelo Pi NoIR, que puede ser conectada directamente a la placa Raspberry Pi utilizada en el cerebro (1). Ésta se usa para capturar, en tiempo real, las imágenes del cultivo. Si bien esta cámara está diseñada para captar imágenes nocturnas (con LED infrarrojo), los resultados de su uso bajo luz artificial son suficientemente buenos como para abarcar las 18 hrs. de luz entregadas.

1.4. Estructura de la memoria

Este documento de memoria consta de 6 capítulos. El Capítulo 2 describe los conceptos básicos involucrados en este trabajo de memoria, los trabajos relacionados y las tecnologías usadas para la implementación de la solución. El Capítulo 3 presenta la estructura general de la solución y sus principales componentes. El Capítulo 4 describe el sistema de control difuso implementado. El Capítulo 5 explica el escenario utilizado para evaluar el sistema, las pruebas realizadas y los resultados obtenidos. Finalmente, el Capítulo 6 presenta las conclusiones y el trabajo a futuro.

Capítulo 2

Antecedentes

En este capítulo se presentan los conceptos básicos involucrados en el trabajo de memoria, así como los sistemas, dispositivos y tecnologías relacionadas con la solución propuesta.

2.1. Conceptos básicos

Para entender las decisiones tomadas al elegir las tecnologías y parámetros aplicados en este sistema, es importante comprender algunos conceptos generales y la aplicación de estos en la agricultura de precisión y sus derivadas:

1. **Internet de las Cosas (IoT)** [13] [37]: Este es un paradigma que describe la interconexión de objetos físicos y aplicaciones/sistemas virtuales, entre los cuales se forma una red de comunicación e intercambio de información, muchas veces inexistente. En la práctica se compone de una serie de dispositivos, librerías, frameworks y protocolos de comunicación con los que se implementan diferentes versiones de este esquema tecnológico. En la mayoría de los casos las soluciones de IoT funcionan con comunicación inalámbrica e inteligente, ajustadas a las necesidades del contexto.
2. **Agricultura 4.0** [22]: Es visualizada como la cuarta revolución industrial, y considera la creciente utilización de componentes electrónicos (sensores, controladores, actuadores, etc.) como apoyo a la agricultura. Estos se envían datos entre ellos, ya sea para comunicar el estado de una variable o para realizar acciones de coordinación. Estos datos son procesados para potenciar la toma de decisiones inteligente (respaldada científicamente) y en tiempo real, o incluso de manera predictiva.
3. **Riego de precisión** [26]: Este concepto busca optimizar (tanto como se pueda) el uso del recurso hídrico, tomando en cuenta características científicas del riego en sí, y la configuración local del cultivo (es decir, red de riego existente, especies a regar, clima, etc.).

De igual forma, existen conceptos asociados al cultivo que sirven para caracterizar los procesos que involucran la dinámica del agua y su influencia en nuestras variables:

1. **Root zone temperature** [23]: Se define como la temperatura del entorno donde se encuentran las raíces de nuestra planta, y tiene una influencia directa en el metabolismo, crecimiento y eficiencia del transporte de nutrientes desde la raíz al resto de la planta.
2. **Water holding capacity (WHC)** [9]: Capacidad de agua que puede retener un sustrato, dependiendo de la textura de suelo (tamaño de partículas de sus componentes), y de la cantidad de materia orgánica presente. El máximo de este rango es denominado *Field Capacity* [36], que se alcanza al regar un sustrato y dejar drenar el exceso. Este es el punto que se busca lograr para realizar un riego óptimo.
3. **Permanent wilting point** [36]: Es el límite tras el cual el sustrato no tiene agua disponible para entregarle a las plantas, por lo que comenzará a morir.
4. **Maximum allowed depletion or deficit (MAD)** [36]: Define el máximo porcentaje de agua que se deja disponible a la planta para absorber tras el riego. En otras palabras, es el máximo de sequedad permitido antes de realizar un nuevo ciclo de irrigación. Se expresa como un porcentaje de la WHC y es posible convertirlo a centímetros. Puede optimizarse según el tipo de cultivo, etapa de crecimiento, cantidad de agua disponible y/o capacidad de bombeo de ésta.
5. **Soil water deficit** [36]: Es la cantidad de agua utilizada por el sistema activo de raíces de la planta. Cuando este valor es igual o mayor que el MAD, debe realizarse el riego hasta llegar al límite óptimo de humedad (*Field Capacity*).
6. **Air porosity** [5]: Es la cantidad de espacio que se crea entre las partículas de un sustrato, lo que permite un mejor drenaje, efectividad en el transporte de nutrientes y oxigenación de la raíces (para la respiración de la planta).
7. **Infiltration Rate** [18]: Es el volumen de flujo de agua que transcurre entre las capas del sustrato por unidad de superficie del suelo.
8. **Evapotranspiración** [15]: Representa la pérdida de humedad de una superficie vegetal, proveniente del proceso de transpiración de la planta en conjunto a la evaporación del agua del sustrato.

La aplicación de estos conceptos se exploran en la siguiente sección a través de algunos ejemplos del mercado y sus características. Esto permitirá que este trabajo de memoria considere algunos factores de diseño que hayan funcionado en otros proyectos de la misma índole.

2.2. Sistemas de monitoreo de cultivos urbanos

El producto más cercano a la solución que se busca desarrollar en este trabajo, es de una compañía chilena llamada *Groupal* [14], cuyo principal producto es el *Groupal One*. Éste es un equipo con pantalla táctil donde se pueden configurar diversos rangos de activación de los actuadores conectados al sistema; por ejemplo, la humedad máxima y mínima para controlar el humidificador. En este equipo también se pueden observar gráficos de las variables de ambiente, configurar rangos horarios para el funcionamiento de los dispositivos, entre otras

funcionalidades. Éste cuenta también con tres enchufes estándar de 220V, para conectar dispositivos asociados a los cuidados de la planta, como el ventilador, el extractor de aire, las luces, etcétera. Además, se incluye el dispositivo Growpal Tank: un estanque con bomba hidráulica para realizar el riego de forma automática, que incluye un sensor de nivel de agua. Growpal One igualmente realiza mediciones de humedad y temperatura ambiental, por medio del dispositivo Growpal Meteo, y de humedad del sustrato, con el dispositivo Growpal Dowser.

Los datos obtenidos desde los sensores son transmitidos a través de tres entradas especiales, localizadas en la parte posterior del equipo. Lo más valioso de esta solución es la creación del estanque autónomo, ideal para cultivos en departamento y oficinas, donde no se cuenta con una salida directa a la llave de agua en el lugar de cultivo. Existen además versiones más complejas y sofisticadas para cultivos en carpas y huertos de exterior, que ubican los distintos componentes especificados de manera optimizada para la comodidad del usuario. El problema principal es la limitación de la cantidad de dispositivos y sensores que se pueden conectar en este sistema, pues en general se tiene más de una planta (y especie), cada una con sus propios requerimientos, y por ende, requiere de sus propios sensores y/o dispositivos. Además, los sensores de Growpal no son inalámbricos, lo cual disminuye considerablemente la comodidad de su uso.

Alternativamente, existe un producto con otro enfoque llamado *Wisegrowth* [35] (también disponible de Chile), que automatiza un máximo de tres dispositivos enchufables de 220V (ventilador, humidificador, luz, etc.). En este proyecto se prioriza la autonomía de los sensores, y la transferencia inalámbrica de los datos generados, además de permitir el control a distancia del equipo y varias funcionalidades por medio de una aplicación móvil. Esta solución cuenta con el equipo cerebro (WiseBOT) y un sensor que mide la humedad/temperatura del ambiente y la humedad del sustrato (WiseSENSE).

En este caso no hay estanque o válvula para el agua, por lo que se considera a este producto como una solución parcial, ya que el riego es la principal preocupación de los cultivadores. Su propuesta de valor está en la ausencia de cables y en su innovadora aplicación de monitoreo, la cual tiene funcionalidades tan variadas como un calendario de cultivo, un menú de configuración de los parámetros que activan el encendido/apagado de los dispositivos conectados, la (des)activación directa de estos, y finalmente la entrega de consejos informativos sobre el cuidado general de las plantas. Los dos ejemplos antes presentados corresponden a un tipo de producto que clasificamos como de *asistencia al cultivo*.

Encontramos también otro tipo de productos, que llamamos de *automatización del cultivo*, como es el caso de *Cloudponics* [7] y *Aspen GrowBox* [2] (disponibles en EEUU). En estos productos la idea o propuesta de valor es mantener la intervención humana a niveles mínimos, y por ende realizan el proceso completo de cultivo de forma automática (desde la germinación, hasta la pre-cosecha). Para ello, se utilizan técnicas de cultivo y cuidado provenientes de investigaciones científicas, específicas a cada especie cultivada. Principalmente, se manejan parámetros óptimos de riego, temperatura, humedad, y acidez del suelo, entre otros. En general, estos equipos tienen forma similar a un refrigerador o clóset, y se debe colocar la semilla en el recipiente de cultivo, indicar qué tipo de planta es, y proveerle acceso a diversas fuentes de recursos requeridos por el sistema, como son el agua, fertilizantes, conexión

eléctrica, etcétera. El principal uso es para cultivo de cannabis, pues tiene mayor valor económico y social, y los tipos de planta corresponden en este caso a las distintas especies/cepas existentes, estudiadas bajo diversas condiciones ambientales.

Ambos tipos de solución (de *automatización* y de *asistencia al cultivo*) cumplen un mismo objetivo; es decir, automatizar de forma parcial o total el proceso de cultivo. Es importante destacar que la solución propuesta tiene que encontrarse en esta misma línea, pero delimitando el grado de autonomía que logrará el sistema, pues es crucial que permita asistir en la mayor parte de los cuidados repetitivos del cultivo, pero permitiendo que cada usuario pueda aplicar sus propias técnicas y conocimiento en todo el proceso de cultivo de la planta. Considerando los costos y los beneficios de las soluciones antes presentadas, se puede ver que éstas dan solución a las necesidades de una parte de los potenciales cultivadores. La principal limitante sigue siendo el elevado costo de estos sistemas, que hace que el cultivo urbano asistido no sea accesible de forma masiva.

Los rangos de precios varían entre los dos tipos de solución. Por un lado, para los productos de *asistencia al cultivo* tenemos un rango de precio alrededor de los \$135.000, mientras que para los productos de *automatización al cultivo* el precio ronda los \$800.000 - \$1.700.000. En ese sentido, el sistema desarrollado en esta memoria tiene una oportunidad de cubrir un nicho de mercado no cubierto por estos otros sistemas, al ofrecer una solución a más bajo costo que las presentadas.

2.3. Thinger.io

Thinger.io [31] es una plataforma para proyectos IoT, que facilita la visualización y guardado de datos, creando una vista preliminar de la dinámica del sistema. Esta misma plataforma permite realizar acciones a través de diversos endpoints, por ejemplo: el envío de mails, mensajes en Telegram y conexión a través de IFTTT (If This, Then That) con servicios como Google Drive, Facebook, Twitter, datos meteorológicos, entre muchos más. IFTTT es un servicio web que permite automatizar tareas y acciones en base a condiciones determinadas por otras aplicaciones o dispositivos, generando rutinas que permiten por ejemplo, publicar en redes sociales, encender/apagar dispositivos de plataformas IoT, enviar mails o mensajes, etcétera.

Aunque esta plataforma tiene grandes prestaciones, en su versión gratuita cuenta con una serie de limitaciones, en términos del número de dispositivos que se pueden conectar, el tamaño mínimo del intervalo de medición, el número de data buckets (campos de almacenamiento), la cantidad de dashboards (gráficos) y endpoints disponibles. El uso de Thinger.io es bastante intuitivo siguiendo sus tutoriales, y para utilizarlo se requiere crear una cuenta para ingresar a la *consola de Thinger.io* [8], desde la cual debemos conectar nuestros dispositivos, incluyendo las librerías asociadas al código del microcontrolador (Arduino IDE).

A pesar de estas limitantes, se conecta el primer prototipo electrónico a esta plataforma, para analizar de forma rápida los valores recolectados a lo largo del tiempo. La Figura 2.1 muestra una captura de la aplicación con estos resultados en una ventana de 16hrs, conteniendo al lado izquierdo los gráficos de las variables consideradas, y a la derecha los valores actuales de dichas variables.

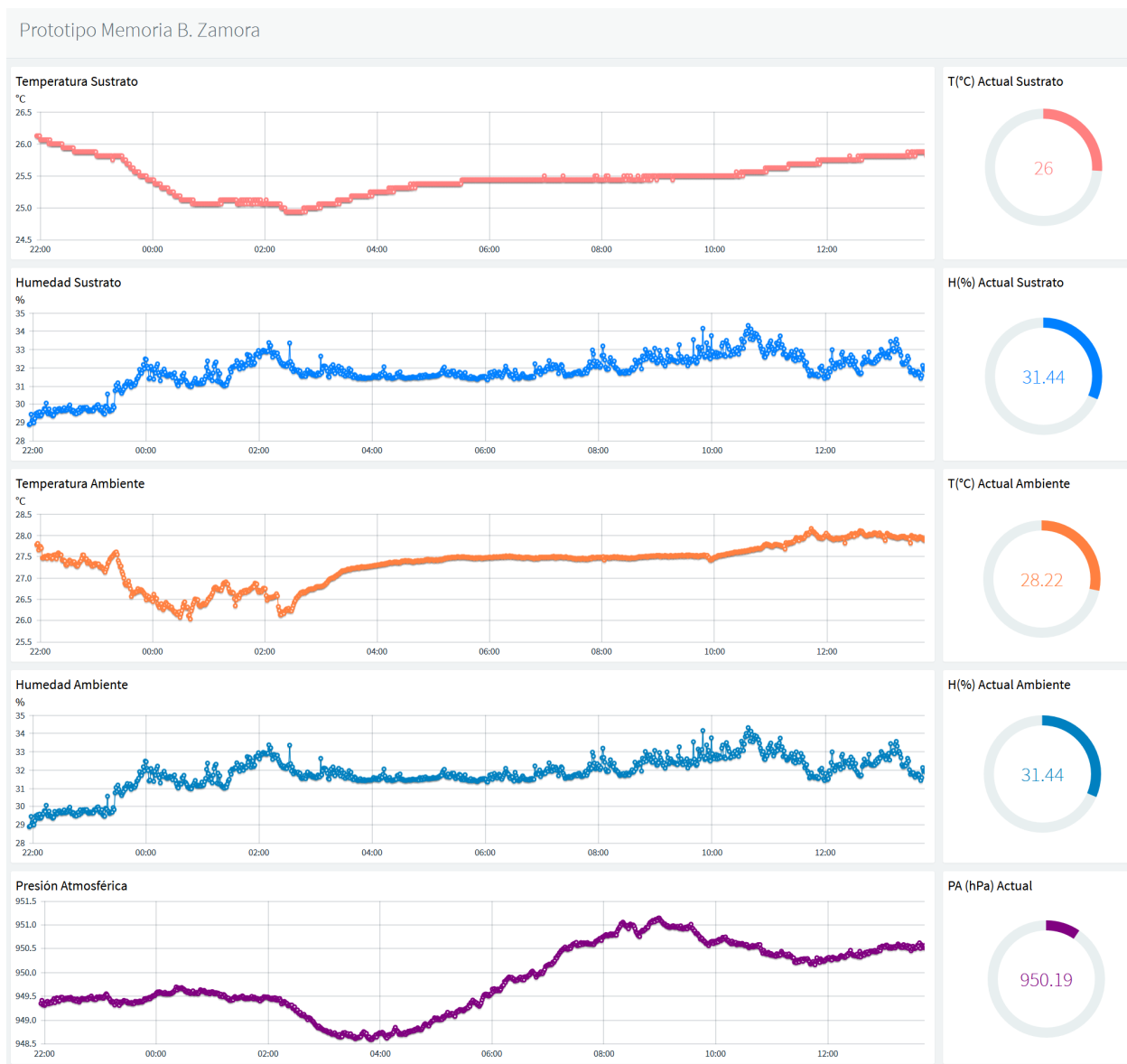


Figura 2.1: Prototipo en plataforma Thingiverse con ventana de 16hrs.

Estos resultados nos permiten observar las variaciones de cada una de las variables a lo largo del día, y a priori se consideran bastante precisos en relación al ecosistema en el cual se sitúa el sistema. Sin embargo, esta plataforma es bastante limitada para conectarse con otros programas o para realizar análisis de los datos, por lo cual se decide explorar en más profundidad otros protocolos de comunicación y programas para construir un sistema más robusto y flexible.

2.4. Message Queue Telemetry Transport

Message Queue Telemetry Transport (MQTT) es un protocolo de comunicación máquina-máquina (M2M), o sea, que está diseñado para la transmisión interna de datos entre dispositivos de una red IoT [3, 28, 10]. La capa de aplicación de este protocolo utiliza TCP/IP como base, estableciendo una conexión entre los dispositivos comunicados y manteniéndola

abierta para su reutilización cada vez que se intercambian nuevos mensajes.

MQTT utiliza el patrón Publisher/Suscriber para permitir la comunicación entre los componentes del sistema; particularmente, el request para la transacción de mensajes es iniciada por el servidor central del sistema (broker). Los clientes se suscriben a tópicos (tipos de eventos) definidos por el broker, siguiendo el patrón antes mencionado. En este proyecto, los clientes recibirán por un lado las variables recolectadas por los sensores, y por otro, los estados de los actuadores entregados por el controlador.

Mosquitto [19] es una implementación particular de MQTT, la cual fue desarrollada por Eclipse. Esta implementación usa un esquema similar a las URLs para representar sus tópicos, creciendo la jerarquía de cada grupo hacia la izquierda; el separador (/) indica cada grupo. Para este trabajo se establecen las siguientes bases para los tópicos del sistema: *ambiente/zona/variable* y *ambiente/zona/actuador*.

Para explicar con más detalle el funcionamiento del modelo de tópicos y el patrón Publisher/Suscriber, se presenta un ejemplo en la Figura 2.2 detallando el proceso de interacciones (en formato de diagrama de secuencia) entre los componentes de la solución propuesta, ligados a la medición de la humedad del sustrato y el control del riego.

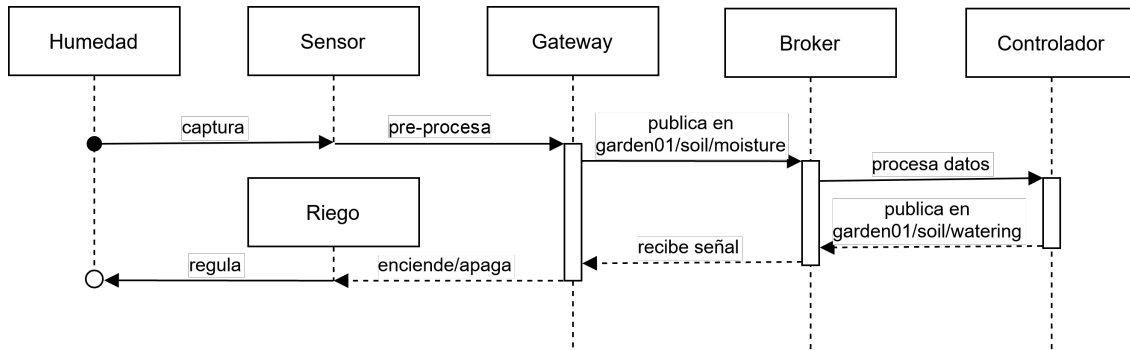


Figura 2.2: Diagrama de secuencia de componentes de humedad y riego del sustrato

A partir de la izquierda superior de la Figura 2.2, observamos el proceso de captura de la humedad desde el sensor, el que a su vez está conectado al gateway. Este último realiza un pre-procesamiento que consiste en el cálculo del promedio de 6 mediciones en un intervalo de 1 minuto. Este valor promedio es publicado en el tópico de la humedad (*garden01/soil/moisture*), que es recibido por el broker y posteriormente enviado a los suscriptores. El controlador es uno de éstos, y lo que hace al recibir cada valor publicado es calcular sus respectivas acciones de control. Dentro del intervalo de decisión (15 minutos) se repite este proceso para finalmente calcular el promedio de los valores y transformarlo a la cantidad de minutos correspondiente. Si este resultado es mayor a cero se escribe el valor de encendido en el tópico de riego (*garden01/soil/watering*), y el broker lo despacha al gateway que está suscrito a este tópico. En base a este valor, envía la señal al relé de la bomba de agua, y una vez pasados los minutos correspondientes el controlador publica el valor de apagado en el tópico de riego para repetir este proceso.

Los mensajes utilizados en Mosquitto son pequeños, y por ende es más eficiente al realizar cada transmisión. Al utilizar TCP/IP podemos definir además el nivel de calidad de servicio

(QoS), que es la cantidad de mensajes que se pierden o repiten en una transmisión. Por último, se destaca la posibilidad de usar SSL/TLS (certificados de seguridad para conexión a Internet) para autenticación o login de forma segura. Las ventajas de este protocolo son su escalabilidad, capacidad de comunicación asíncrona, bajo acoplamiento entre clientes, bajo consumo energético (por su ligereza), uso mínimo de ancho de banda y robustez en la transmisión de mensajes.

Para utilizar Mosquitto y conectarlo con otros programas, principalmente diseñados en Python dentro de la Raspberry Pi, requerimos utilizar algunas librerías y una base de datos para generar este puente entre los componentes de nuestro sistema, los cuales son presentados a continuación.

2.5. Paho MQTT y InfluxDB

Eclipse Paho MQTT [11] es una librería que permite el diseño del broker y clientes de Mosquitto, entregando funciones simplificadas para cada uno de los procesos requeridos (publicación, suscripción, formato de tópicos, etc.). En particular, se utiliza la librería para el desarrollo del broker, que recibe y transmite tanto los valores sensados desde el gateway (NodeMCU), como las acciones a realizar desde el controlador difuso (*Scikit Fuzzy* [24]). Además, se utiliza para el desarrollo del data collector, que guarda toda la información transmitida en una base de datos de serie de tiempo (TSDB) llamada *InfluxDB*, especial para proyectos IoT.

Esta base de datos entrega un estilo de consulta similar a SQL, con una estructura centrada en el tiempo como identificador de los datos, compuesta además de medidas, series y puntos. Cada punto consiste en varios pares llave-valor (llamados *fieldset* y *timestamp*), que se agrupan usando un *tagset* para así definir una serie. Cuando estas series son agrupadas por un string identificador, forman una medición o métrica. Cada punto entonces puede ser indexado por el tiempo, o alguno de sus *tagset*.

En el caso de esta memoria, las tablas convencionales corresponden a cada variable sensada, y los *tagset* representan los ambientes y zonas. Con estas dos tecnologías tendremos la base para un procesamiento de datos rápido y de alta disponibilidad, tanto para el almacenamiento como para la transmisión de estos. El único recurso pendiente es la visualización de estos datos, para lo cual se introduce el siguiente framework.

2.6. Grafana

Grafana [12], es un software Open Source que permite la visualización de métricas utilizando como fuente, bases de datos de varios tipos. En este trabajo de memoria se ocupa la colección descrita en la sección anterior (InfluxDB). Grafana permite la visualización de varios tipos de gráficos (barras, líneas, geográficos, etc.), además de actualizaciones en tiempo real de sus valores. Por otra parte, este software cuenta con funcionalidades potencialmente útiles, como por ejemplo, conexiones a distintos tipos de endpoints, el establecimiento de alertas, límites visuales en los gráficos, administración de cuentas, y una serie de componentes que se exploran en detalle en el Capítulo 5.

Se selecciona el software para este proyecto principalmente por la flexibilidad que entrega para el análisis de los datos y su visualización. La estética del sistema es profesional y permite una serie de cambios de estilo que lo hacen adaptable a cualquier tipo de proyecto. Además que las tecnologías seleccionadas con anterioridad (Mosquitto e InfluxDB) son de fácil integración al sistema.

Una de las funcionalidades requeridas para la plataforma es el envío de notificaciones y alertas, por lo que se requiere la implementación de la siguiente tecnología.

2.7. BotFather

Esta herramienta desarrollada por Telegram [30], es utilizada para el diseño y administración de *bots* que funcionan dentro de esta aplicación de mensajería. Estos *bots* son aplicaciones desarrolladas por terceros, con las que se interactúa a través de mensajes, comandos y requests dentro de la aplicación de mensajería Telegram, para luego recibir varios tipos de respuestas, como por ejemplo, notificaciones, noticias, mails y también formas más complejas de interacción como son los servicios de música, juegos, compras, entre otras.

En este trabajo de memoria, se envían notificaciones y alertas de forma automática, requiriendo únicamente el ingreso del comando de inicio del bot en el dispositivo móvil del usuario. Esta acción de enviar mensajes es gatillada por una funcionalidad del framework Grafana, que permite establecer alertas ante algún cambio en las condiciones de las variables, en concreto, cuando se alcanzan puntos críticos o si ocurre algún cambio en el estado de los actuadores.

2.8. Sistema de Control Difuso y Scikit Fuzzy

Un sistema de control o controlador se describe como un software y/o un conjunto de dispositivos (eléctricos, hidráulicos, etc.) cuya función es controlar, gestionar o regular las métricas de otro sistema, que de alguna forma (conocida o no) impactan en su dinámica [21]. Esto se realiza para ordenar o direccionar estas variables con un objetivo específico, usualmente definido por expertos en el uso del sistema, como por ejemplo, para mantener un nivel de agua fijo en un estanque o una temperatura específica para reactores químicos y nucleares. Los controladores se componen de al menos tres elementos básicos:

1. Variable o métrica a controlar
2. Actuador
3. Set-point o valor de referencia

El controlador recibe como input el estado de una o varias métricas y lo compara con su valor de referencia (óptimo). Utilizando este resultado, calcula una salida correspondiente a la acción de control, que representa la intensidad o la cantidad de tiempo que deben encenderse los actuadores respectivos del sistema, para así controlar esta variable y mantenerla lo más cerca posible de este set-point. Respecto a la complejidad del proceso de cálculo de la acción, podemos encontrar dos clasificaciones: 1) sistemas de control clásico o convencionales, y 2)

sistemas de control óptimo o inteligente.

El primer grupo describe un conjunto de controladores cuyo método es más bien simple, tales como los que se indican a continuación:

1. ON/OFF, que realiza la acción de encender o apagar un dispositivo, por ejemplo, cuando existe un valor de una variable fuera de rango.
2. Controlador proporcional, integral y derivativo (PID), que a través del cálculo del error (diferencia entre el valor actual y el de referencia de una variable), realiza transformaciones matemáticas para establecer la magnitud con la que se debe ajustar un proceso de control (actuador)
3. Sistemas a lazo abierto, que reciben una medición de la variable a controlar, pero cuya salida es independiente de ésta; es decir, no se utiliza este valor para ajustar la acción de control.

El segundo grupo de controladores comprende una serie de técnicas modernas para resolver el problema de control, entre las que se destacan los siguientes ejemplos: inteligencia artificial, redes neuronales, control predictivo, y control difuso, entre otros [1].

En esta memoria, se utiliza este último tipo de control (difuso) explicado en detalle en el Capítulo 4. La implementación es realizada en el cerebro (Raspberry Pi 3B+). Existen muchas librerías para la implementación de este tipo de sistemas de control, sin embargo, aquella que permite mayor flexibilidad y personalización es *Scikit Fuzzy* [25], desarrollada por la comunidad SciPy, que cuenta con larga trayectoria en el desarrollo de software científico, además de una gran comunidad en línea. La librería permite resolver una serie de problemas típicos, como la clusterización, y el diseño de sistemas simples o mixtos de control difuso.

2.9. Especificaciones técnicas de componentes del sistema

A continuación se presentan diversos componentes que corresponden a los ya indicados en este documento, tales como sensores, actuadores, controladores, etcétera. Estos dispositivos fueron considerados como potenciales candidatos a ser utilizados en el sistema desarrollado, luego de realizar un análisis de las alternativas disponibles en portales de e-Commerce electrónicos nacionales y extranjeros. Se indican además los componentes escogidos, que fueron seleccionados en base a su calidad, precio y disponibilidad para ser incorporados en los tiempos definidos para el proyecto. Los precios indicados corresponden a la fecha de Marzo del 2021 y el link a cada producto se encuentran en el nombre del dispositivo.

2.9.1. Placas de desarrollo

Como se detalla en la Sección 1.3, se necesitan dos tipos de placas para el proyecto, la primera de éstas se utiliza para el desarrollo del cerebro del sistema. Ésta debe ser capaz de procesar los datos provenientes del gateway y del controlador difuso, y comunicarse con ellos vía WiFi, además de alojar tanto la base de datos como el software de visualización. Por ende, esta placa debe ser capaz de correr varios programas simultáneamente, y contar

con herramientas de apoyo al debugging de estos programas. Con estos requisitos, se hace prácticamente directa la elección de la Raspberry Pi, facilitando además la incorporación de una cámara para la visualización de imágenes en tiempo real, con el módulo Pi NoIR (ya adquirido con anterioridad). En la Tabla 2.1 se presentan las alternativas exploradas.



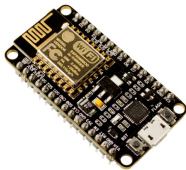
Nombre	Descripción	Foto referencial
Raspberry Pi 3B+	Cuenta con un procesador quad-core de 64 bits con 1,4 GHz, red inalámbrica de banda dual 2.4/5 GHz y Bluetooth 4.2 / BLE. Además tiene 4 puertos USB 2.0 y 40 pines GPIO (sin conversión ADC). Precio: \$43.990	
Raspberry Pi 4	Incluye un procesador quad-core de 64 bits a 1,5 GHz, hasta 8GB de RAM, LAN inalámbrica de banda dual 2.4 / 5.0 GHz, Bluetooth 5.0, Gigabit Ethernet, 2 USB 3.0 y 2 USB 2.0, además de 40 pines GPIO (sin conversión ADC). Precio: \$79.990	

Tabla 2.1: Placas de desarrollo para la implementación del cerebro

Es clara la superioridad de la Raspberry Pi 4 en la mayoría de sus componentes. Sin embargo, se decide trabajar con la Raspberry Pi 3B+ debido a que ya se posee una, sumado a que sus características son suficientes para el nivel de procesamiento requerido en el sistema propuesto.

Para el gateway, se necesita otra placa con la capacidad de comunicarse inalámbricamente (sin módulos extra), procesar bajas cantidades de información (4 sensores cada 10 segundos, más la recepción de mensajes de Mosquitto) y compatibilidad con varios tipos de transmisión de datos desde los sensores y hacia los actuadores (digital, analógica, serial, etc.). Una diferencia clave entre las alternativas presentadas, son los protocolos de comunicación base para los cuales fueron diseñadas cada placa. La Tabla 2.2 presenta estas alternativas.

Nombre	Descripción	Foto referencial
NodeMCU v3 ESP8266 (WiFi)	Cuenta con un módulo WiFi de 2.4 GHz integrado, un pin análogo y 17 digitales, basado en el SoC ESP8266. Soporta programación OTA (vía WiFi). Precio: \$3.981	

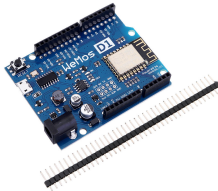




<p>WeMos D1 R2 V2.1.0 (WiFi)</p>	<p>Basada en ESP-8266EX. Se puede programar con Arduino IDE. Tiene 11 pines digitales y 1 pin de entrada Análoga. Soporta programación OTA (vía WiFi). Precio: \$9.890</p>	
<p>Arduino MKR1000 (WiFi)</p>	<p>Basado en el microcontrolador ATSAMW25, cuenta con un módulo WiFi WINC1500 de 2,4 GHz integrado, 7 pines análogos, 8 digitales, 12 salidas PWM y una salida analógica. Precio: \$25.000</p>	
<p>Arduino MKRFOX1200 (SigFox)</p>	<p>Posee un módulo ATA8520 SigFox para conexión LPWAN y un microcontrolador SAMD21 que gestiona la interfaz GPIO. Tiene 15 pines digitales y 7 analógicos, 12 pines PWM y 8 con interrupciones externas. Precio: \$101.000</p>	
<p>Arduino MKR WAN1300 (LoRa)</p>	<p>Basada en el Atmel SAM21 y el módulo LoRa Murata CMWX1ZZABZ, cuenta con 8 pines digitales, 7 pines análogos de entrada y 1 de salida. También con 12 pines PWM y 8 de interrupción externa. Precio: \$44.990</p>	
<p>Arduino MKR GSM1400 (GSM)</p>	<p>Está basada en el microcontrolador SAMD21 y el módulo GSM ARAU201. Cuenta con 8 pines digitales, 7 entradas analógicas y 1 salida analógica. 12 pines PWM y 8 de interrupción externa. Precio: \$121.417</p>	

Tabla 2.2: Placas de desarrollo para la implementación del gateway





La primera restricción es el precio de las placas Arduino, que son excesivamente altos para el tipo de solución que busca abordar el proyecto. Hay que considerar que la mayoría de estas placas son diseñadas para apoyar soluciones que tienen otro tipo de requerimientos, como por ejemplo la distancia de transmisión, conexión satelital, etcétera.







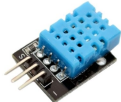
En el caso de este sistema, al considerarse un cultivo a lo más de tamaño medio (invernadero), con distancias cortas y sin interferencias entre los componentes del sistema, se decide hacer uso de las placas basadas en red WiFi; se decidió esto también por la cantidad de datos

que deben transmitirse. Considerando estos aspectos, se escoge la solución más accesible y de mayor trayectoria en el mercado (y comunidad de soporte), que es la placa NodeMCU ESP8266 v3. Para incorporar los sensores a esta placa, necesitamos simplemente una entrada analógica para lectura, y el resto digitales, por lo que los pines existentes son suficientes para lo requerido en este proyecto.

2.9.2. Sensores

En la Tabla 2.3 se presenta el listado completo de sensores explorados, incluyendo variables que escapan al alcance de esta memoria. Los sensores analizados capturan alguna de las variables clave para optimizar el desarrollo de cualquier tipo de cultivo. Faltan además una serie de alternativas de uso industrial, que permiten lecturas más precisas y de mayor calidad, pero de las cuales no se pudo conseguir más información (sobre precios y otros detalles).

Nombre	Descripción	Foto referencial
Sensor de humedad FC-28 (YL-69 y YL-38)	Sensor de humedad medido por medio de la resistencia entre las dos patas. Precio: \$1200	
Sensor de humedad FC-28 (recubierto en oro)	Equivalente al anterior, pero con un recubrimiento de oro para evitar la corrosión a largo plazo. Precio: \$2.300	
Sensor de humedad sellado	Permite evitar igualmente la corrosión por medio de un sellado con plástico. Precio: \$5.990	
Sensor de humedad capacitivo v1.2	Mide la humedad por medio del cambio de la capacitancia en la placa. Precio: \$2.691	

<p>Teros 12</p>	<p>Detección de humedad en nivel más profesional (en los materiales de construcción y en la capacidad de detección). Precio: \$394.530</p>	
<p>Vegetronix VH-400</p>	<p>Detección de humedad para nivel semi-profesional. Precio: \$28.500</p>	
<p>Stevens Water HydraGO</p>	<p>Se conecta a Android/Apple vía bluetooth con la app HydraMon. Detecta humedad, temperatura, conductividad y permitividad dieléctrica, que se registra en conjunto a la fecha y hora, y localización GPS. La data se guarda en un archivo CSV</p>	
<p>Sensor de temperatura DS18B20</p>	<p>Realiza mediciones de temperatura en suelo o agua, con rangos de -55°C a 125°C. Precio: \$2.990</p>	
<p>Sensor de agua</p>	<p>Detecta niveles de agua, principalmente para niveles de agua lluvia, con rangos de 10 % a 90 % de humedad. Precio: \$2.489</p>	
<p>Xiaomi Mi Flora</p>	<p>Por medio de la aplicación Flower Care se conecta a través de bluetooth y entrega información sobre nutrientes, temperatura, luz y humedad. Precio: \$18.000</p>	
<p>Sensor de humedad y temperatura ambiente DHT11</p>	<p>Mide temperatura y humedad ambiente. Los rangos son de 0°C a 50°C, 20 % a 80 % de humedad y 1Hz sampling rate. Precio: \$2.390</p>	

<p>Sensor de humedad y temperatura ambiente DHT22</p>	<p>Mide temperatura y humedad ambiente. Los rangos son de -40°C a 125°C, 0 % a 100 % de humedad y 0.5Hz sampling rate. Precio: \$4.349</p>	
<p>Sensor de humedad y temperatura ambiente BME280</p>	<p>Mide temperatura, humedad ambiente y presión. Los rangos son de -40°C a 85°C, 0 % a 100 % de humedad relativa y 1Hz sampling rate. Precio: \$2.980</p>	
<p>Sensor de pH Crowtail (Elecrow)</p>	<p>Sensor pH que obtiene resultados en menos de 2 minutos. Rango de temperatura es de 0°C a 60°C. Precio: \$20.932</p>	
<p>Sensor de sonda pH BNC</p>	<p>Equivalente al sensor anterior, pero realiza la medición analógica a través de una sonda. Rango de temperatura es de 0°C a 80°C. Precio: \$13.387</p>	
<p>Sensor Keystudio CCS811</p>	<p>Detecta monóxido de carbono y calidad de aire. Precio: \$11.990</p>	
<p>Sensor CO2 MG-811</p>	<p>Sensor de alta calidad con un rango de detección de 400 a 10000ppm de CO2. Precio: \$59.990</p>	




SparkFun Weather Shield	Detecta presión barométrica, humedad relativa, luminosidad y temperatura. También permite conexiones extras para agregar sensores de velocidad y dirección de viento, lluvia y GPS. Precio: \$39.990	
Sensor MH-714A	Módulo infrarrojo para medición de CO2. Precio: \$46.498	
Gravity TDS Meter	Permite la detección de la conductividad del agua. Precio: \$79.990	

Tabla 2.3: Sensores analizados

Para la humedad de sustrato se toma la decisión de escoger el sensor capacitivo, pues su precio (\$2.691) es bastante razonable, cuenta con buena resolución y es ampliamente utilizado en proyectos comerciales. Además, los sensores de resistencia son frecuentemente criticados por la corrosión y contaminación que generan a largo plazo en nuestras plantas. Para la temperatura de suelo se encuentra únicamente el sensor DS18B20 tipo sonda para el rango de precios manejado. Finalmente para la humedad y temperatura ambiente se selecciona el sensor BME280, pues sus características son bastante buenas para el trabajo requerido, además que las pruebas encontradas a largo plazo [17] que comparan este con otros sensores similares, dejan ver que esta es la opción de mayor fiabilidad.

2.9.3. Componentes escogidos para la solución

En la Tabla 2.4 se presentan las especificaciones de los sensores escogidos obtenidas desde los datasheet de cada uno de ellos. De esta manera se puede diseñar el sistema considerando el porcentaje de error que estos sensores presentan, además del tiempo de muestreo máximo necesario para establecer el intervalo de medición del controlador.

Sensor	Tiempo de muestreo	Error
BME280	1 Hz	3% Humedad 1°C Temperatura
Capacitive Soil Moisture Sensor	Instantánea	6%
DSB18B20	0.75 Hz	0.5°C

Tabla 2.4: Especificaciones técnicas de los sensores del sistema

2.9.4. Costo de la solución

El costo total del sistema es la suma de los componentes seleccionados con anterioridad, y el detalle de este cálculo es presentado en la Tabla 2.5.

Producto	Precio
Raspberry Pi 3B+	\$43.990
Cámara PI NoIR	\$32.990
NodeMCU v3 ESP8266	\$3.981
Sensor capacitivo de humedad v1.2	\$2.691
Sensor DSB18B20	\$2.990
Sensor BME-280	\$2.980
Total	\$89.662

Tabla 2.5: Costo agregado del sistema propuesto

Como se observa, el principal factor de costo son dos dispositivos: la Raspberry Pi 3B+ y la Cámara Pi NoIR. Los costos de ambos dispositivos pueden ser reducidos, encontrando mejores ofertas en tiendas electrónicas en el extranjero. La cámara en particular tiene un sobre-costo que no entrega utilidad al sistema, dado que está diseñada para utilizar en escenas nocturnas, por lo que podría considerarse la alternativa de usar la versión tradicional de la cámara que es el modelo PiCamera v1.3, con un precio alrededor de los \$15.000.

Otro punto importante es la comparación de este costo total con los valores de los otros productos presentados en la Sección 2.2. Para esto se presenta la Tabla 2.6 que permite esta comparación con los precios encontrados a la fecha de Marzo del 2021.

Claramente el precio del sistema no es comparable con el de las soluciones de automatización del cultivo, y el público objetivo difiere principalmente por el poder adquisitivo de éste. Sin embargo, con respecto a las dos primeras soluciones de asistencia, se logra una reducción considerable de costos, esto sin considerar claramente el diseño e implementación del encapsulamiento del sistema. Por lo tanto, con respecto a los componentes tenemos un resultado positivo, considerando igualmente las posibles reducciones de costo mencionadas anteriormente. Otro punto importante es que en el caso de la solución aquí reportada, ésta

Producto	Precio
GrowPal One - KIT	\$150.000
Wisegrowth	\$120.000
Cloudponics	\$1.763.876
Aspen Grobox	\$830.612
Sistema propuesto	\$89.662

Tabla 2.6: Comparación productos del mercado con el sistema propuesto

considera el uso de una cámara que no está presente en las alternativas analizadas; o sea, es un feature adicional de la solución propuesta.

2.10. Resumen

En este capítulo se presentaron diversas soluciones que se venden empaquetadas para apoyar el proceso de cultivo urbano, y se dio una referencia del costo de dichos sistemas. Además, se hizo una revisión de los componentes que típicamente se utilizan para implementar estos sistemas, y se presentó una referencia del costo de esos componentes. Finalmente, se indicó cuáles son las tecnologías, protocolos, modelos y componentes que se escogieron para implementar el prototipo reportado en esta memoria, y su costo aproximado.

En el próximo capítulo se describe más en detalle la solución propuesta.

Capítulo 3

Propuesta de Solución

En este capítulo se presentan los principales requisitos de la solución, así como su estructura y el soporte de datos. El sistema propuesto consiste en elementos de hardware y software, siendo los primeros la base para la recolección de datos, comunicación y ejecución de las acciones de control, mientras que el segundo entrega la lógica para la toma de decisiones automatizadas, las herramientas para notificar al usuario sobre situaciones en su cultivo y las plataformas de visualización/monitoreo. El detalle de cada componente y sus funcionalidades se presenta a continuación.

3.1. Principales requisitos de la Solución

Para especificar las características de la solución, se definen los siguientes requisitos funcionales, que abarcan los pilares esenciales para un sistema de cultivo semi-automatizado. Asegurando estos elementos lograremos cubrir las necesidades básicas de los usuarios objetivo de este proyecto, y cada funcionalidad incorporada en el futuro representará mejoras o correcciones que harán del sistema flexible ante la evolución de las tecnologías y los requerimientos de los distintos cultivadores que lo utilicen.

1. El sistema permitirá la integración de sensores de temperatura y humedad del sustrato y del ambiente (medidores), así como también los sistema de ventilación y riego (actuadores).
2. El sistema realizará el control automático con el fin de mantener las variables del ecosistema dentro de los rangos especificados.
3. El sistema permitirá la personalización de los parámetros, rangos y valores de referencia (representativos de la especie cultivada), según las necesidades y experiencia del usuario.
4. El sistema registrará los datos temporales de las variables monitoreadas, manteniendo la evolución histórica de los valores de las mismas.
5. El sistema entregará visualizaciones de valores históricos y en tiempo real de las variables, para que el usuario observe la dinámica del sistema de forma gráfica, y pueda

evaluar el comportamiento de la solución en base a esa información.

6. El sistema notificará al usuario por medio de mensajes en Telegram bajo dos consignas: alertas sobre situaciones críticas de las métricas y cambios de estado en actuadores.

Es importante agregar a lo antes mencionado que se pretende que este sistema sea de bajo costo, de manera que pueda estar disponible para una gran cantidad de personas. En ese sentido, los dispositivos utilizados como parte de la solución también deben ser accesibles.

3.2. Arquitectura lógica

La arquitectura física descrita en la Sección 1.3 nos entrega una idea general del diseño de la arquitectura lógica del sistema, dado que las funcionalidades y métodos de comunicación presentados (véase Figura 1.1) deben implementarse con tecnologías diseñadas para este fin, agrupadas en entidades lógicas que cumplen con un rol específico dentro del sistema. En la Figura 3.1 se muestran estos componentes, donde las flechas simbolizan el flujo de información. Se indica con una línea sólida los datos provenientes desde los sensores, y con una línea punteada las acciones de control dirigidas hacia los actuadores.

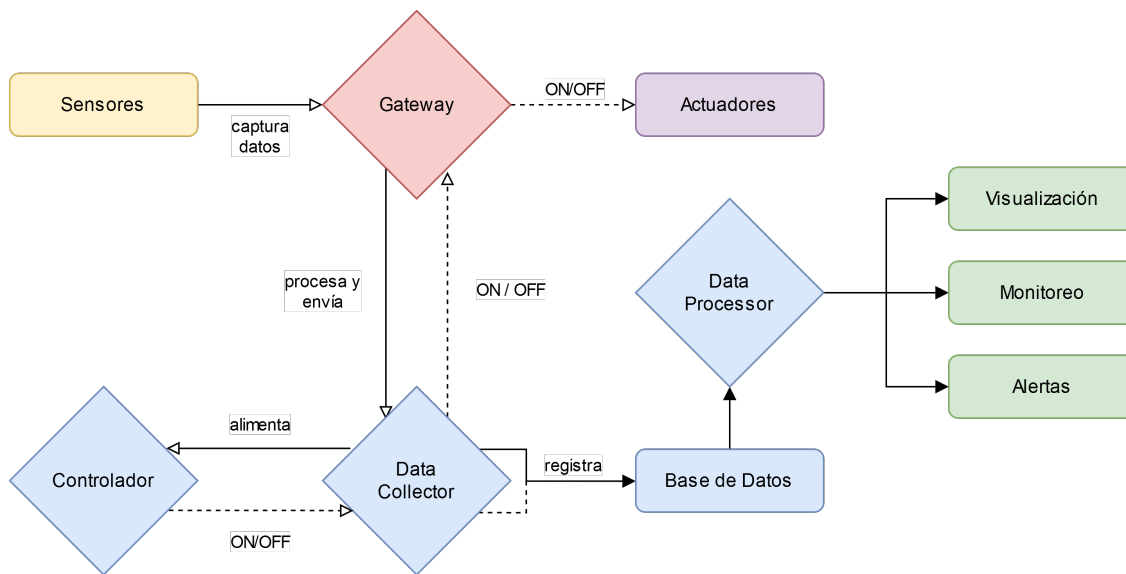


Figura 3.1: Arquitectura lógica del sistema

Las tecnologías seleccionadas para esta memoria son las que se observan en la Figura 3.2, representando una instancia de esta arquitectura lógica dado que como se explica en el Capítulo 2, existen tecnologías alternativas para cada componente propuesto. Es importante notar que este modelo de solución corresponde a un sistema basado en la red WiFi a nivel hogar, por lo que se requiere una conexión local para su funcionamiento.

Comenzando por la izquierda de la Figura 3.2, se tiene el flujo de datos proveniente desde los sensores, los cuales están conectados al gateway (NodeMCU), que actúa como publicador dentro de Mosquitto al escribir estos datos en su respectivo tópico, asignado por el broker de la red. Este último está alojado en el data collector (Raspberry Pi), el cual al recibir esta información la procesa entregando los datos al controlador difuso (localizado en

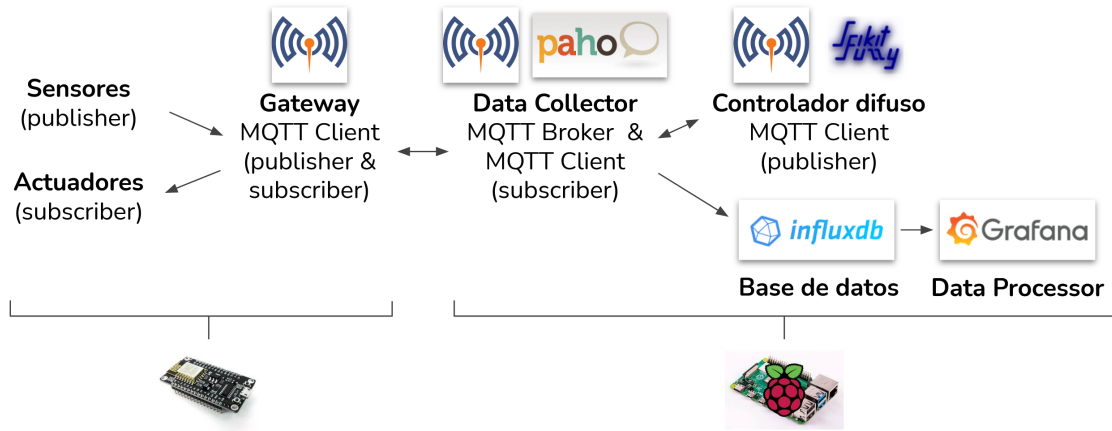


Figura 3.2: Instancia de la arquitectura lógica del sistema propuesto

el mismo equipo) que está implementado en Scikit Fuzzy. Luego del intervalo de decisión del controlador (descrito en la Sección 4.2), entrega una acción de control que se escribe en el tópico correspondiente. Esta acción está dirigida a un actuador conectado de forma directa (relé) o inalámbrica (Smart Plug) al gateway, que está suscrito al tópico que define el estado (ON/OFF) de cada actuador, tal como fue ilustrado en la Figura 2.2.

En paralelo a este proceso de control, el Data Collector actúa como suscriptor de los tópicos de los sensores y actuadores del sistema. Éste registra dicha información en la base de datos InfluxDB que se utiliza además como fuente para la generación de visualizaciones en el software Grafana. El sistema requiere contar con el envío de notificaciones, para que el usuario sepa qué está ocurriendo con su cultivo, además de tomar acciones en caso de requerir intervención o identificar algún problema con el sistema. Para realizar esto, dentro del mismo software existe una funcionalidad que se explica en detalle en el Capítulo 5.

Con esta arquitectura lógica, tendremos los componentes del sistema que agrupan las funciones y clases desarrolladas en este trabajo de memoria. Para explicar el detalle de esta propuesta y la interacción entre los elementos del sistema, se explica a continuación la arquitectura del software.

3.3. Arquitectura del software implementado

Para comprender cómo se sitúa el sistema propuesto en relación a los usuarios y otros sistemas, se presentan los siguientes esquemas bajo el modelo C4 [32], que facilita la comprensión de la arquitectura del software implementado, representado a través de diferentes niveles de abstracción. Estos niveles van desde el de mayor abstracción (contexto general), hasta el detalle mismo del código de cada componente, y son presentados en este mismo orden en esta sección.

3.3.1. Contexto general del software implementado

En la Figura 3.3 se muestra la estructura del sistema desarrollado, representado a través del primer nivel de abstracción. Este modelo muestra (en color azul) los usuarios y los sub-

sistemas considerados en esta memoria, en conjunto con los sistemas propuestos en la memoria del Sr. Rodrigo Soria [27] (en color rojo).

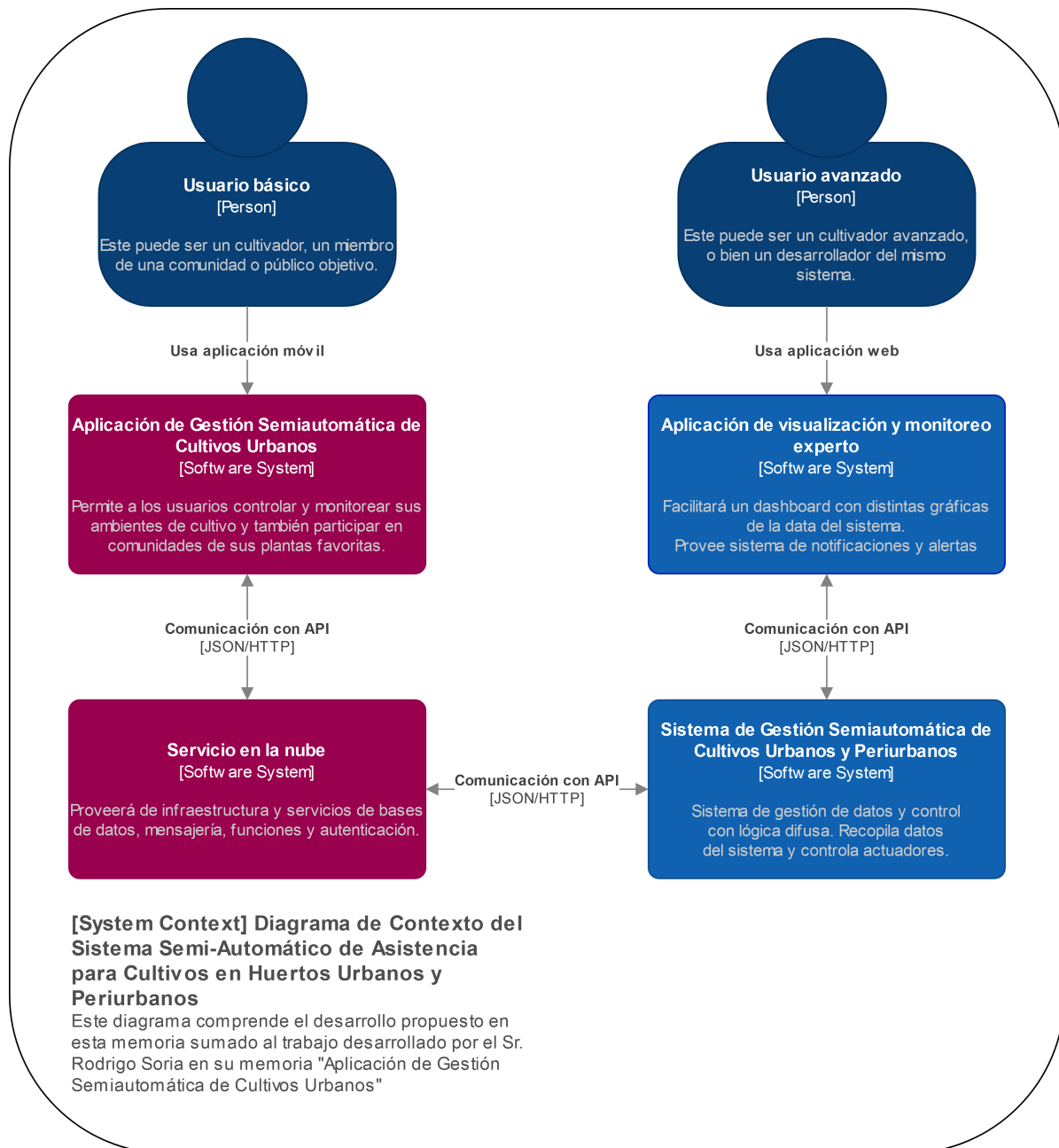


Figura 3.3: Diagrama de contexto del sistema

Tal como se aprecia en la Figura 3.3, el proyecto está dividido en dos sub-sistemas (o módulos): 1) el módulo de visualización y monitoreo experto, y 2) el módulo de gestión semi-automática de cultivos. Cada uno se compone de servicios basados en las tecnologías descritas con anterioridad, y el detalle es explorado en la siguiente sub-sección.

3.3.2. Contenedores del software implementado

En el siguiente nivel del modelo C4, los sub-sistemas son llamados contenedores, y se describen como módulos que corren de forma independiente pero conectados entre sí, para lograr el funcionamiento del sistema en su totalidad. Las Figuras 3.4 y 3.5 representan la estructura de los sub-sistemas antes mencionados, utilizando el segundo nivel de abstracción del modelo C4.

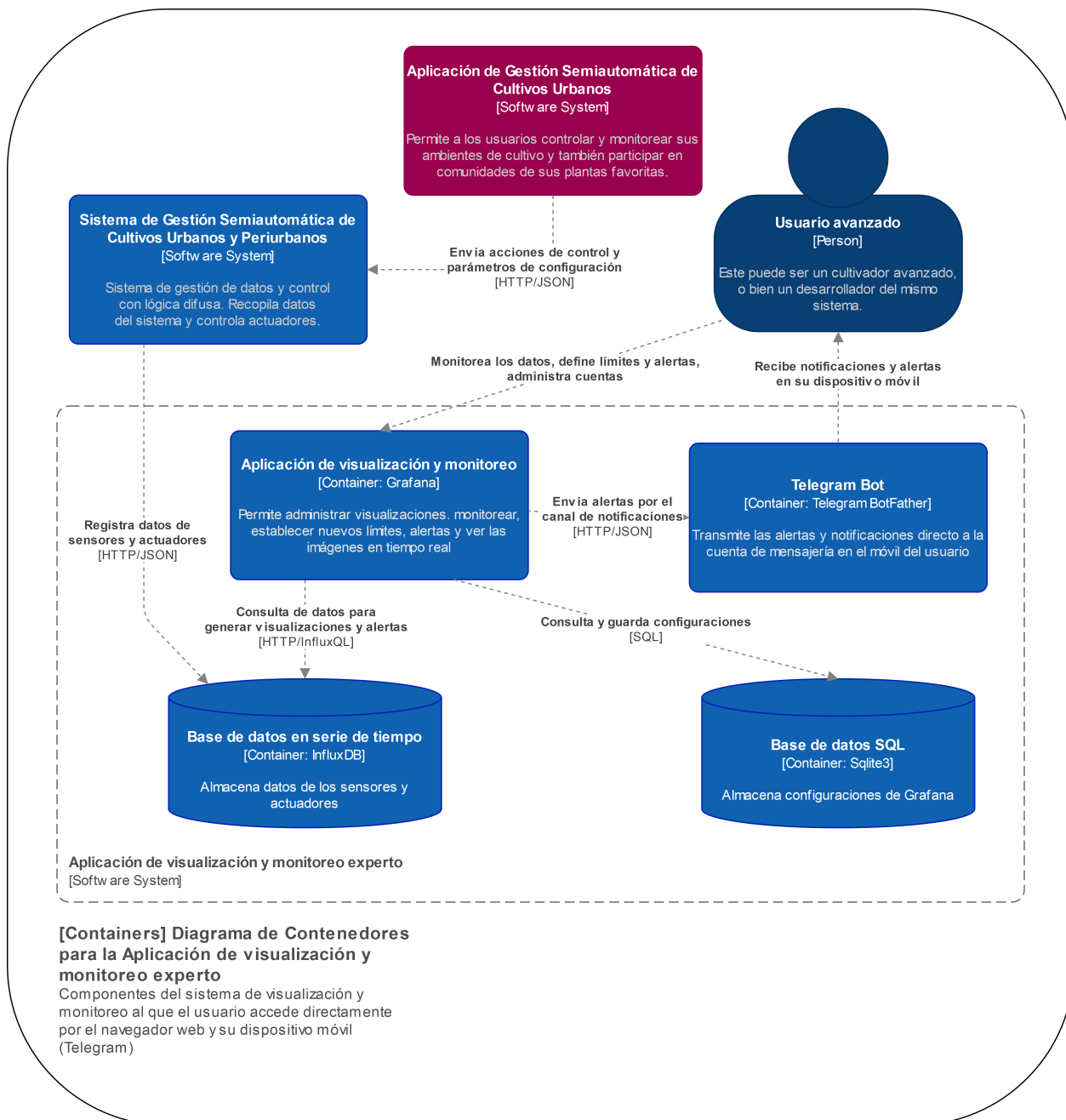


Figura 3.4: Diagrama de contenedores del módulo de visualización y monitoreo experto

Este primer módulo (Figura 3.4) integra diversos contenedores, que abarcan por un lado las bases de datos necesarias para guardar las configuraciones y valores del sistema, y por otro lado los programas que sirven de interfaz para el usuario, que en este caso son la aplicación

de visualización y monitoreo (Grafana) y el bot de notificaciones y alertas (BotFather de Telegram).

Cada uno de estos interactúa entre sí a través de acciones representadas con líneas punteadas, que especifican igualmente las tecnologías involucradas (de ser necesario). Un ciclo completo involucra primero el registro de los datos de sensores y actuadores en la base de datos InfluxDB (provenientes del Sistema de Gestión Semiautomática), los que constantemente son consultados por Grafana para generar los gráficos de cada métrica. El usuario se conecta a esta plataforma vía web para observar los datos históricos, y además recibe las notificaciones en su dispositivo móvil en la aplicación Telegram a partir de las alertas definidas por el desarrollador.

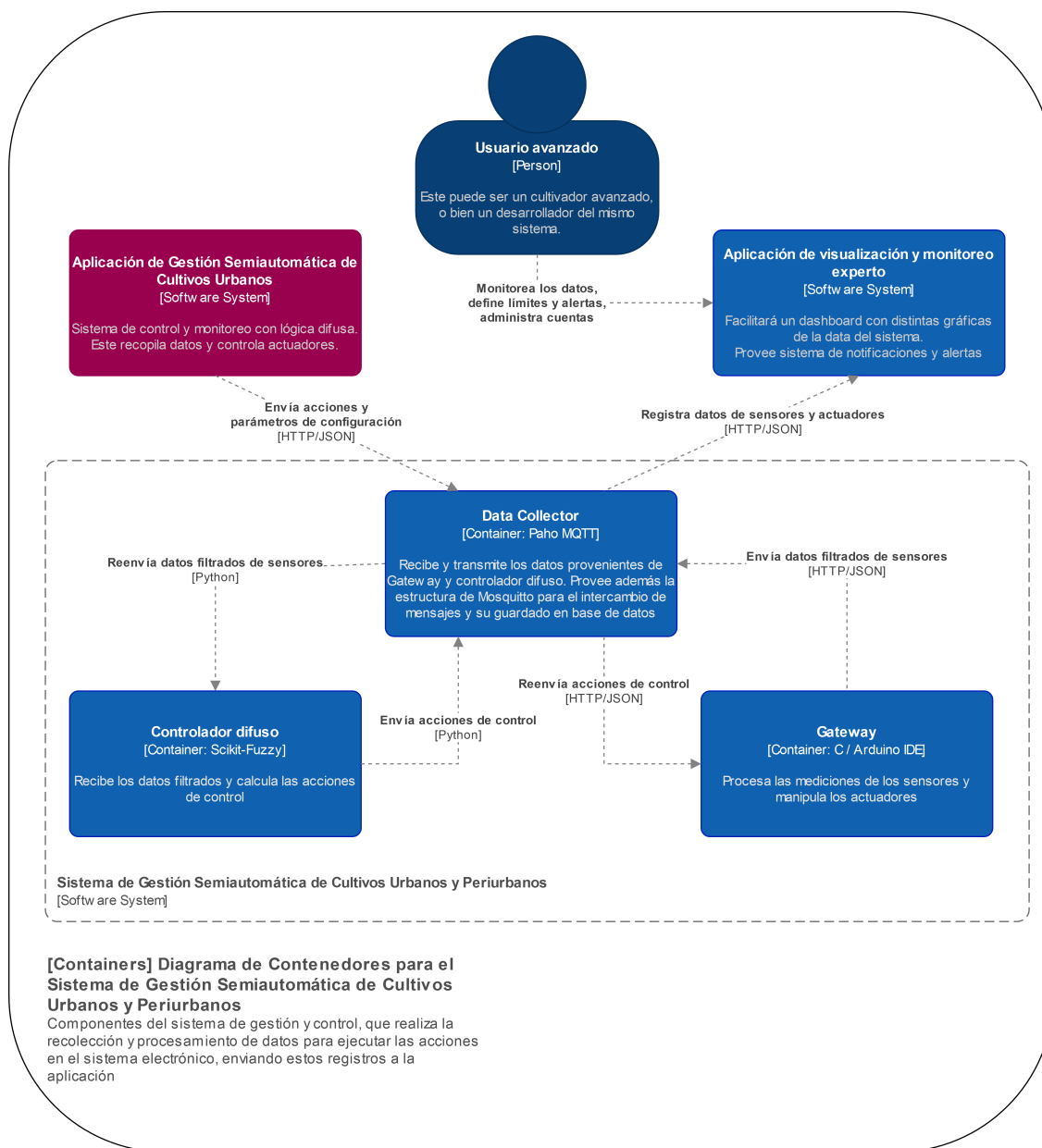


Figura 3.5: Diagrama de contenedores del módulo de gestión semi-automática de cultivos

Este segundo módulo (Figura 3.5) muestra el detalle de los contenedores del Sistema de

Gestión Semiautomática, que involucra toda la parte electrónica, de comunicación y procesamiento de datos. Por un lado tenemos el Data Collector que recibe y transmite todos los datos del sistema, y funciona como servidor central de esta información. Por otra parte tenemos el Gateway que se instala en el ambiente de cultivo para recibir los datos de los sensores y ejecutar las acciones de control. Finalmente contamos con el controlador difuso que recibe los datos promediados escritos en los tópicos de cada variable para calcular las acciones de control, y al completar el intervalo de decisión, escribir finalmente las acciones de control en sus tópicos respectivos. Cada uno de estos contenedores corre su propio programa en base a las tecnologías especificadas bajo el nombre, compuestas de librerías y dispositivos electrónicos configurados para este fin, las cuales se presentan en la siguiente sub-sección.

3.3.3. Componentes del módulo de visualización y monitoreo

En la Figura 3.6 se observa el tercer nivel del modelo C4, que entrega el detalle de cada contenedor, formado por varios elementos denominados componentes. Estos componentes representan un conjunto de funcionalidades (controladores, patrones, etc.) encapsuladas en un mismo entorno de ejecución (o sea, el del contenedor). En este caso, el único contenedor que necesitamos describir con más detalle para comprender la dinámica interna del sistema, es el del módulo de visualización y monitoreo, ya que los otros son simplemente un único archivo de código y serán explicados en el siguiente nivel.

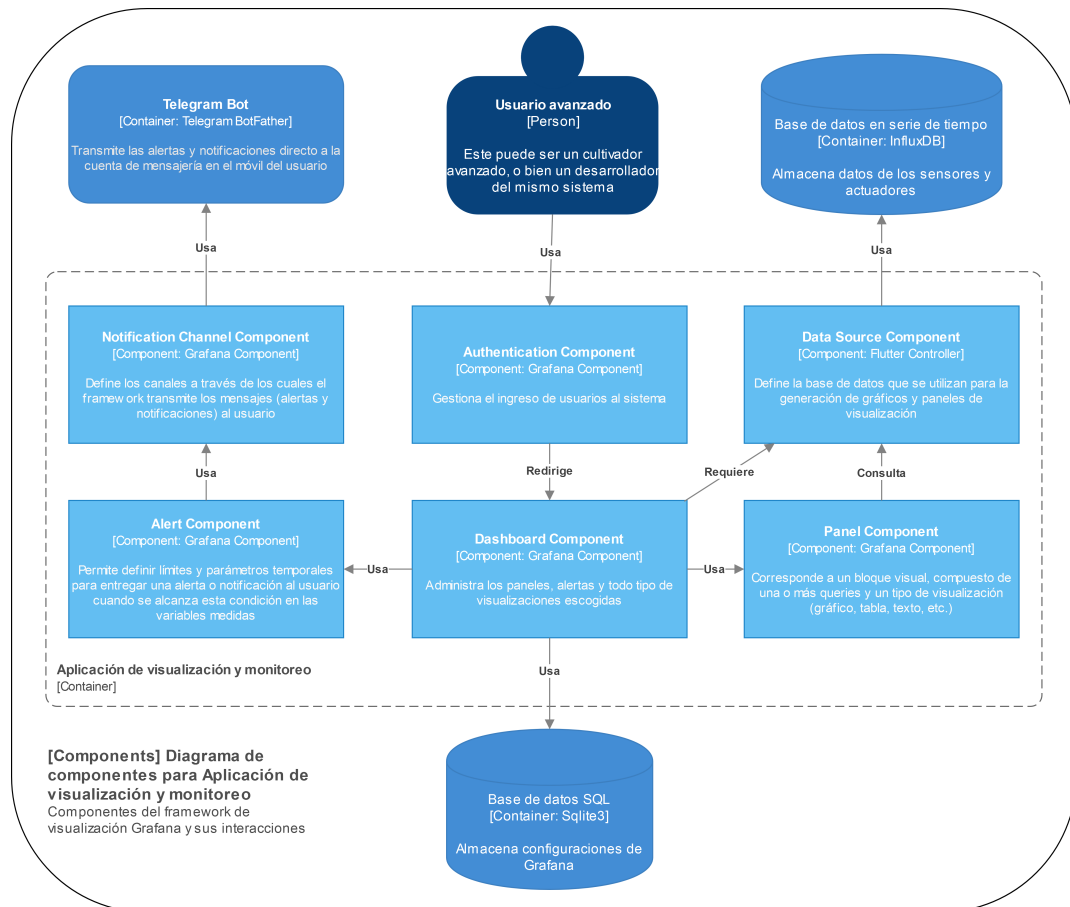


Figura 3.6: Diagrama de componentes del módulo de visualización y monitoreo

3.4. Modelo de datos del sistema

El cuarto y último nivel del modelo C4 se utiliza para presentar el código de cada componente, empleando para esto diagramas UML, ya sean estos diagramas de clases, componentes u otro tipo de artefacto de software. Sin embargo, ese nivel de detalle escapa a los objetivos de esta sección, y en su lugar se utiliza un diagrama de entidad-relación para mostrar una representación de los datos generados en el sistema. La Figura 3.7 nos muestra este esquema, separando las entidades en grupos (color celeste) que se corresponden con los componentes descritos en la arquitectura lógica (Figura 3.1).

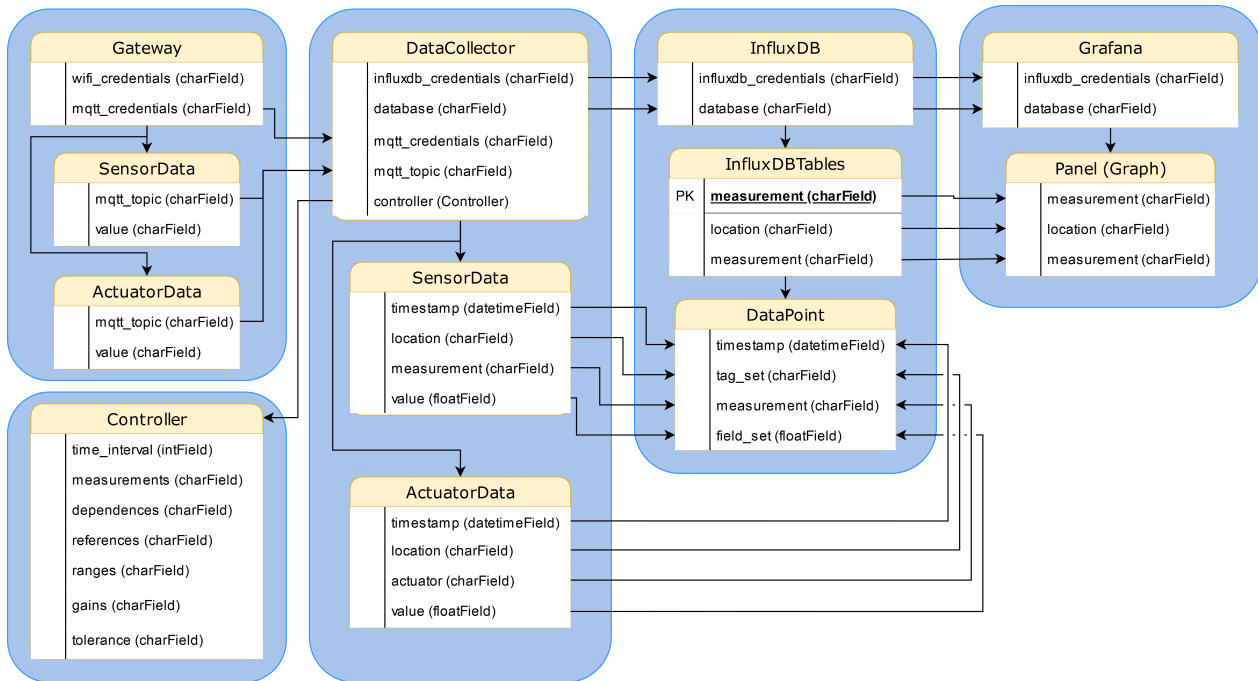


Figura 3.7: Modelo de datos del sistema desarrollado

En la parte superior izquierda de la Figura 3.7 se observa el conjunto de entidades definidas en el gateway. Para la configuración de este dispositivo, se requieren las credenciales de la red WiFi del hogar (`wifi_credentials`) y las de Mosquitto (`mqtt_credentials`), necesarias para publicar y suscribirse a los mensajes de la red. Ambas credenciales son incluidas directamente en el código, pero idealmente deben ser entregadas vía programación OTA (Over-The-Air) u otro tipo de comunicación inalámbrica, con el fin de mejorar la seguridad y la escalabilidad de estos sistemas. En el mismo dispositivo se generan y reciben dos tipos de datos: `SensorData` y `ActuatorData`, compuestos por un tópic (`mqtt_topic`) específico para cada variable o actuador (con la base presentada en la Sección 2.4), y un valor (`value`) correspondiente a la medición o el estado del actuador.

Bajo este grupo encontramos el controlador difuso, que es una instancia de la clase `Controller` creada en el `DataCollector`, el cual se encarga de entregarle los datos provenientes del gateway para que este los procese y entregue las acciones de control correspondientes. La forma de personalizar los resultados de control es a través de los atributos que se indican a continuación:

1. `time_interval` es el tamaño del intervalo de decisión medido en minutos.

2. *measurements* y *dependences* son las variables y los actuadores con sus respectivas dependencias en formato de diccionario, aunque estas variables son fijas en el código, dado que las reglas deben definirse de forma manual.
3. *references* y *ranges* son los valores definidos en la Sección 4.4 para caracterizar el comportamiento generalizado de la especie a plantar.
4. *gains* son las ganancias de cada variable para definir una ponderación para su acción de control resultante
5. *tolerance* es el paso utilizado para generar los conjuntos difusos del delta del error de las variables

En la parte central de la Figura 3.7, podemos observar el *DataCollector* y los datos generados por éste. En sus atributos contamos con la información relacionada a la base de datos, que son las credenciales de escritura (*influxdb_credentials*) y el nombre de la base de datos (*database*). Luego tenemos las credenciales de Mosquitto (*mqtt_credentials*) y los tópicos disponibles (*mqtt_topic*). Las credenciales son especificadas en el archivo de configuración de Mosquitto, alojado en el mismo dispositivo (Raspberry Pi). Los tópicos, en cambio, son especificados en el código mismo del data collector, y definen la base para el resto de los elementos de la red.

Para escalar este proyecto en función de ampliar la cantidad de ambientes y plantas, se debe automatizar el proceso de generación de tópicos. En este mismo sentido, la instancia única del controlador difuso (*controller*) del *DataCollector* tendría que ser flexible a las preferencias de cada usuario, ya sea generando varias instancias de éste, o permitiendo modificar los atributos de la clase (*ranges* y *references* y las reglas) en tiempo de ejecución. Estas propuestas no son exploradas en esta memoria.

Los datos publicados en los tópicos serán recibidos por el callback de Paho MQTT, que es la tecnología base del data collector, a partir de los cuales se realiza un proceso de parsing para así transformarlos al formato de dos clases del tipo *NamedTuples*, una para actuadores (*ActuatorData*) y otra para sensores (*SensorData*). Cada uno de estos datos cuenta con un atributo temporal (*timestamp*), y otros asociados al tópico base (*location* y *actuator/measurement*), además del valor mismo de la variable o estado del actuador (*value*).

Al lado derecho de la Figura 3.7, tenemos las entidades relacionadas a la base de datos que incluyen tanto las credenciales de acceso como el nombre de la base de datos (*influxdb_credentials* y *database*). Para fines ilustrativos, se muestra cómo se agruparía la información en formato relacional con la tabla *InfluxDBTables*, que tiene como nombre alguna de las variables o actuadores, y el resto de atributos como tags para las queries.

Para entender la realidad tras este proceso, se incluye bajo esta entidad el formato real de ingreso de datos, utilizando el protocolo en línea de InfluxDB. Cada valor ingresado bajo este formato llamado *DataPoint*, consiste en un string con la siguiente sintaxis: `"/measurement/tag_set/field_set/timestamp/".` Estos datos son generados a partir de las entidades *ActuatorData* y *SensorData* del *DataCollector*, cuyos atributos se convierten al formato JSON para el posterior procesamiento en base de datos.

El último grupo representa las entidades asociadas a la plataforma de visualización. Para acceder a las visualizaciones se requieren las credenciales de lectura para la base de datos (*influxdb_credentials*) y el nombre de ésta (*database*). El framework Grafana hace uso de diversos paneles en los dashboards, y cada uno representa un tipo de visualización (gráficos, estadísticas y otros) que es generada a partir de consultas realizadas a la base de datos. Cada consulta consta de los mismos atributos que los incluidos en las *InfluxDBTables*. Podemos incorporar distintas variables, ambientes u otro tipo de tags y campos en una misma consulta, además de filtrar o modificar por medio de funciones a través del lenguaje InfluxQL.

3.5. Discusión

En este capítulo se presentaron los principales requisitos de la solución, así como la arquitectura lógica y la arquitectura de software del sistema implementado. Allí se discute también las potenciales capacidades del sistema debido a la arquitectura que implementa. Además, como parte del diseño del sistema se presenta su modelo de datos.

La arquitectura reportada muestra la forma en que se agrupan e interoperan los diversos componentes del sistema. Esto también permite ver la forma en la cual el sistema puede ser extendido y/o modificado.

El actor principal en esta arquitectura es el controlador (o cerebro) del sistema. En el próximo capítulo se presenta el sistema de control difuso que constituye, en gran medida, el comportamiento de dicho componente.

Capítulo 4

Sistema de Control Difuso

En este capítulo se describen los principales componentes del sistema de control difuso, así como los mecanismos que éste implementa para llevar a cabo la regulación de las variables. Este sistema considera solo una parte de las variables relevantes de un cultivo en huertos urbanos y periurbanos, sin embargo, sirve para demostrar el funcionamiento de la lógica difusa en un sistema semi-automatizado de asistencia. Además, se incorpora la medición de métricas para evaluar el rendimiento de los métodos de control.

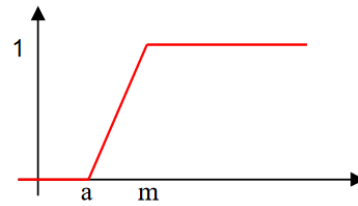
4.1. Introducción a los controladores difusos

La lógica difusa es un tipo de lógica que permite extender la concepción clásica de unos y ceros para definir pertenencia a un conjunto, incorporando un elemento llamado función de pertenencia, que representa un grado más humano de incertidumbre o ponderación en la definición de conjuntos. Matemáticamente, se representan como pares ordenados que indican el valor del elemento y su grado de pertenencia al conjunto difuso, de la forma: $A = \{(x, \mu_A(x)) / x \in X\}$

En otras palabras, un conjunto difuso permite definir cualquier variable en términos lingüísticos, como por ejemplo cuando clasificamos un objeto con respecto a su temperatura (muy frío, un poco caliente, etc.). Con este tipo de conjuntos podemos especificar aún más estas definiciones, afirmando que un objeto está caliente cuando su temperatura se encuentra cerca de los 70°C ($\mu_{caliente}(70^\circ\text{C}) = 1$), y alejándose (disminuyendo $\mu_{caliente}$) de esta clasificación hacia ambos extremos, que es cuando el objeto se vuelve frío (10°C) o cuando se vuelve muy caliente (100°C). Algunos ejemplos de las funciones más utilizadas para representar esta pertenencia en los conjuntos difusos son las presentadas en la Figura 4.1.

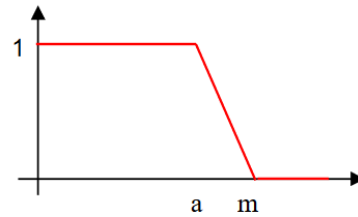
- Función Gamma (Γ)

$$\mu(x) = \begin{cases} 0 & x \leq a \\ \frac{x-a}{m-a} & a < x < m \\ 1 & x \geq m \end{cases}$$



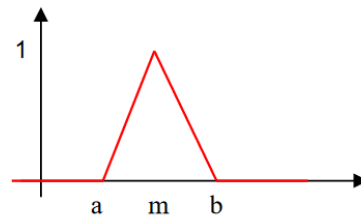
- Función L: se puede definir como

$$\mu(x) = 1 - \Gamma$$



- Función Lambda o Triangular

$$\mu(x) = \begin{cases} 0 & x \leq a \\ \frac{x-a}{m-a} & a < x \leq m \\ \frac{b-x}{b-m} & m < x \leq b \\ 0 & x > b \end{cases}$$



- Función PI o Trapezoidal

$$\mu(x) = \begin{cases} 0 & x \leq a \\ \frac{x-a}{b-a} & a < x \leq b \\ 1 & b < x \leq c \\ \frac{d-x}{d-c} & c < x \leq d \\ 0 & x > d \end{cases}$$

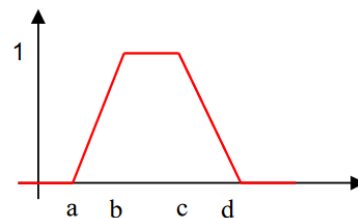


Figura 4.1: Funciones de pertenencia clásicas [29]

Un tipo de modelo difuso clásico es el de Mamdani [20], que consiste en una estructura lingüística basada en reglas heurísticas, donde cada variable de entrada (u) y salida (y) es definida por conjuntos difusos. Podemos apreciar una representación de los componentes de este tipo de sistemas de lógica difusa (y sus interacciones) en la Figura 4.2, detallando a continuación la funcionalidad de cada elemento

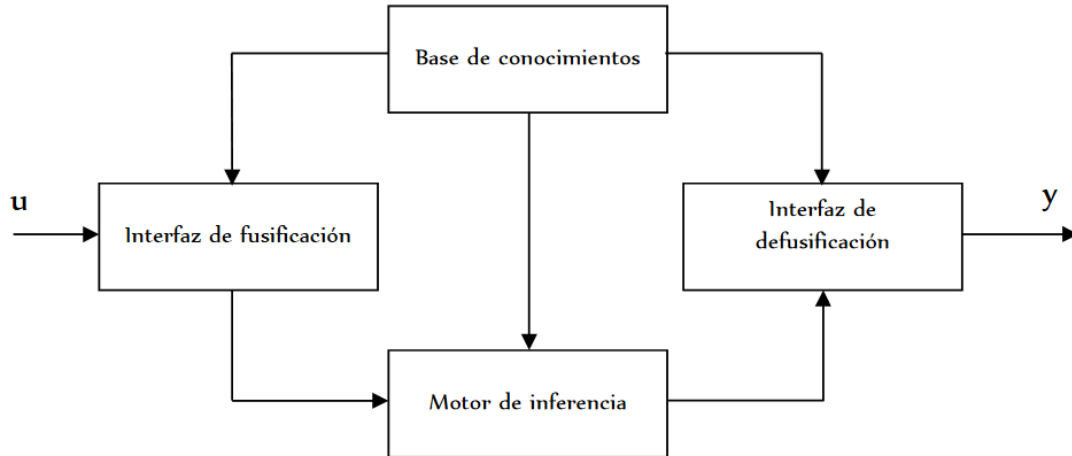


Figura 4.2: Estructura de un modelo difuso tipo Mamdani [29]

1. **Interfaz de fusificación:** Realiza la transformación de la variable de entrada (u), al modelo de conjuntos difusos, con sus funciones de pertenencia respectivas para la variable.
2. **Base de conocimientos:** Contiene las funciones de pertenencia y las reglas difusas, que son del estilo: "Si u_1 es A y u_2 es B entonces y es C ", con A, B, C conjuntos difusos de las respectivas variables. Estas reglas pueden provenir de la definición de expertos o a través del modelamiento matemático del sistema.
3. **Motor de inferencia:** Calcula la variable de salida, utilizando para esto el proceso de inferencia difusa. Se puede resumir en tres pasos:
 - (a) Cálculo de la activación de cada regla: $W_i = \text{mín}(\mu_{A_i}, \mu_{B_i})$, utilizando el mínimo como intersección de los conjuntos difusos de las entradas.
 - (b) Cálculo de la activación de la consecuencia de cada regla, intersectando esta con la activación del primer paso: $\mu_{C'_i} = \text{mín}(W_i, \mu_{C_i})$.
 - (c) Evaluación de la unión de reglas para obtener un conjunto difuso de salida, utilizando para esto el máximo: $\mu_{C'} = \text{máx}(\mu_{C'_i}) \forall i = 1, \dots, n$.

La utilización de mín y máx está relacionada con las operaciones básicas de lógica difusa que no son exploradas en esta explicación.

4. **Interfaz de defusificación:** Entrega una salida discreta y determinista a partir del conjunto C' de salida, utilizando para esto un método de defusificación (máximo, centro de gravedad, bisección, entre otros) presentados en la Figura 4.3.

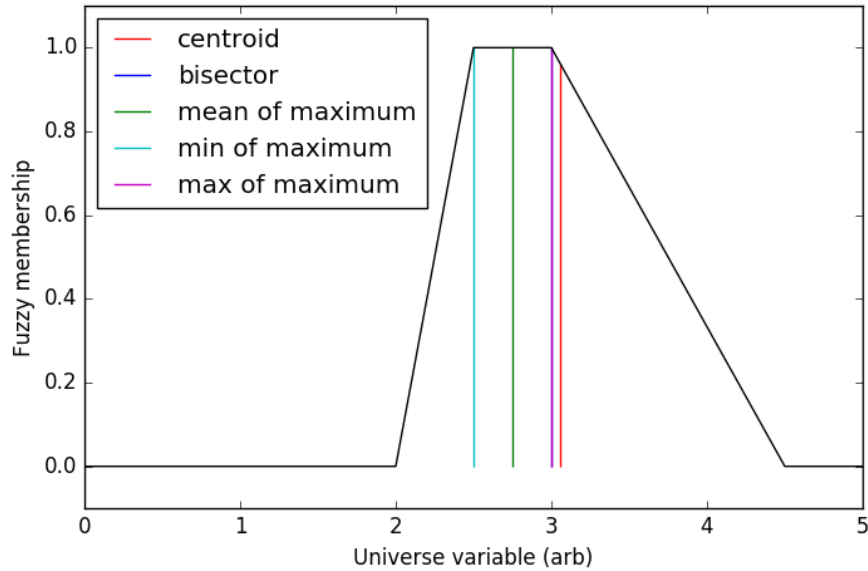


Figura 4.3: Métodos de defusificación [24]

Con esta base podemos diseñar varios controladores difusos [6] [4], como por ejemplo los PI difusos tradicionales (PIF) (también llamados error-delta), que se componen de un único controlador difuso que procesa dos entradas (error y su incremento) por medio del sistema de lógica difusa tipo Mamdani presentada previamente, como se observa en la Figura 4.4; los PI-F difusos (PI-F), que utilizan dos controladores difusos que pre-procesan las entradas y un controlador PI para la obtención de la acción de control final; los self-tuning tipo-PI difuso (STPIF), que utilizan un controlador difuso para obtener la acción de control y otro para ajustar un parámetro asociado a la ganancia de la acción de control; y muchas otras combinaciones posibles. En general este diseño está sujeto a las necesidades de cada problema y la complejidad del proceso a controlar. Para el caso de esta memoria, nos basta con un PI difuso tradicional, dado que el objetivo es evaluar la utilidad de un sistema de automatización con las características presentadas.

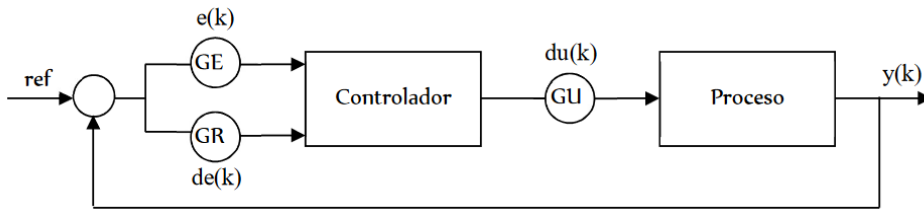


Figura 4.4: Estructura estándar de un controlador PI difuso [29]

Este controlador es un sistema en lazo cerrado, dado que utiliza la variable de estado $y(k)$ (resultado de aplicar la acción de control $u(k)$ al proceso) para calcular el error $e(k+1)$ y el delta del error $de(k+1)$ (variación) a partir del valor de referencia ref con las fórmulas 4.1 y 4.2. Con estos valores se alimenta un sistema de lógica difusa tipo Mamdani, definiendo cada componente de este según los requerimientos del problema a resolver. Las otras variables GE, GR, GU son llamadas ganancias, y se utilizan como ponderación de cada variable del controlador, para así ajustar los resultados hasta obtener el comportamiento deseado.

Corresponde ahora definir los componentes detallados para construir el controlador tipo error-delta propuesto en esta memoria, por lo tanto se comenzará explicando las variables que son utilizadas por este sistema en la siguiente sección.

4.2. Variables consideradas

La dinámica del ecosistema de nuestras plantas se ve afectada por una serie de factores ambientales y/o humanos. Muchos de estos tienen uno o más métodos de control y regulación (por ejemplo, humedad y riego) denominados actuadores, los cuales influyen de forma proporcional o inversa al valor de la variable correspondiente. En esta memoria se considera solamente cuatro:

1. **Temperatura de Sustrato (TS):** Representa la temperatura local del sustrato utilizado para la planta, que es el medio donde reside la raíz (tierra, fibra de coco, etc.).
2. **Humedad de Sustrato (HS):** Representa la humedad local del sustrato utilizado para la planta.
3. **Temperatura Ambiente (TA):** Representa la temperatura ambiental del entorno de cultivo, tanto en interior como en exterior.
4. **Humedad Ambiente (HA):** Representa la humedad relativa del ambiente que, en otras palabras, es la relación entre la cantidad de vapor de agua que contiene una masa de aire y la máxima que ésta podría contener.

Estas variables, en conjunto a la concentración de CO₂ y de nutrientes (fertilización), son los valores más importantes a tomar en cuenta para el desarrollo ideal de un cultivo. Para lograr el control deseado se necesita definir la especie a cultivar, sus características y cuidados. La traducción de esto al sistema de control propuesto consiste en encontrar los valores de referencia (set-points) para cada variable, y observar cómo se comporta el ecosistema de la planta en comparación a estos, para así ejecutar las consecuentes acciones de regulación y control. Las variables de referencia y sus acrónimos son:

1. **Temperatura de Sustrato referencial (TSref)**
2. **Humedad de Sustrato referencial (HSref)**
3. **Temperatura Ambiental referencial (TAref)**
4. **Humedad Ambiental referencial (HAref)**

Podemos encontrar estos valores en la literatura o consultarlos con expertos en el cultivo de la especie escogida. Con estos valores de referencia, el controlador realiza el cálculo del error en cada paso, que indica cuán alejadas están nuestras variables del óptimo. Además se calcula un delta del error que indica su crecimiento. El sistema diseñado es del tipo controlador error-delta y recibe como entrada ambos valores en un ciclo cerrado para retro-alimentar los resultados obtenidos. Las fórmulas para el cálculo de ambas variables son las siguientes, con V correspondiente a la variable o métrica y t el paso temporal:

$$error(V, t) = V_{ref} - V(t) \quad (4.1)$$

$$delta(V, t) = error(V, t) - error(V, t - 1) \quad (4.2)$$

El principal objetivo es controlar de manera eficiente las variables seleccionadas. Para esto, contamos con los siguientes actuadores, que tienen una influencia directa en estas:

1. **Ventilación** Disminuye los valores de humedad y temperatura ambiente.
2. **Riego** Aumenta los valores de humedad de sustrato y disminuye los valores de temperatura de sustrato.

Finalmente, se definen algunas constantes necesarias para el proceso de cálculo de las acciones de control, que involucra desde las mediciones capturadas en los sensores hasta la ejecución de la acción final de control:

1. **Intervalo de medición:** Corresponde al tiempo durante el cual el gateway (y sus sensores) realizan las mediciones, utilizando para esto un método de agregación (promedio) que entrega un valor representativo del intervalo.
2. **Número de mediciones:** Corresponde a la cantidad de mediciones que el gateway realiza por cada sensor en el intervalo de medición. Se debe cuidar que este número sea menor al tiempo de muestreo máximo del sensor, indicado en el datasheet de cada proveedor (Tabla 2.4).
3. **Intervalo de decisión:** Corresponde al tiempo durante el cual el controlador difuso procesa los valores de medición filtrados, para obtener acciones de control y luego realizar un promedio de estas y entregar una acción de control final representativa del conjunto, que será ejecutada durante el próximo intervalo de decisión.

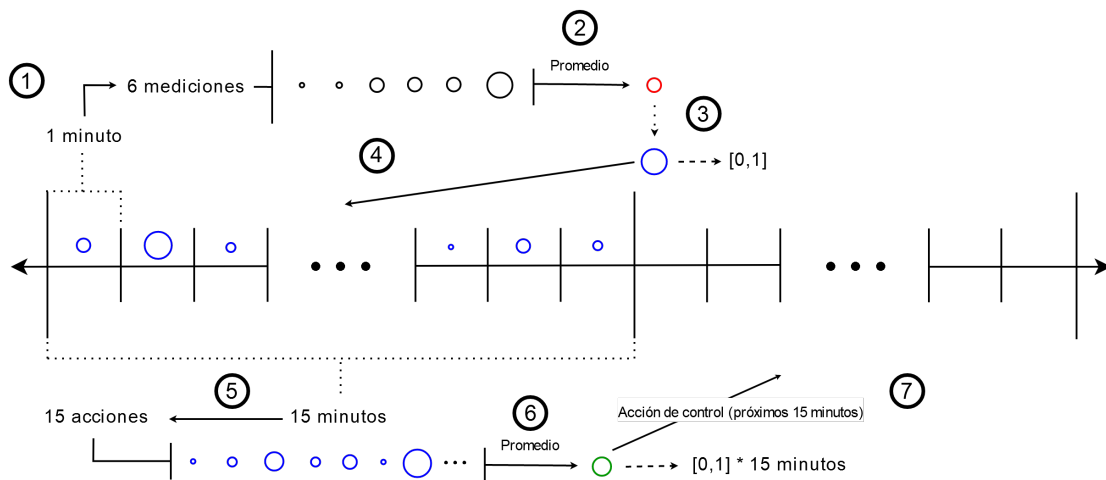


Figura 4.5: Proceso de cálculo de una acción de control

La Figura 4.5 muestra gráficamente este proceso de cálculo utilizando los valores reales del sistema, detallando a continuación cada paso de este proceso:

1. Con un intervalo de medición igual a 1 minuto, el microcontrolador (NodeMCU) realiza una medición cada 10 segundos (6 por minuto) que guarda en un buffer temporal.
2. De las mediciones anteriores, se calcula el promedio para así entregar un valor representativo y reducir los posibles outliers o errores. Puede explorarse el uso de otro tipo de filtros para esta etapa.
3. Con el promedio calculado, se publica este valor y luego el controlador (Raspberry Pi) recibe, procesa y calcula su respectiva acción de control.
4. Cada acción de control calculada por minuto es guardada en una lista temporal en la memoria local del controlador.
5. Al alcanzar el fin del intervalo de decisión (15 minutos), se llama a la función de agregación con los valores guardados.
6. Se observa que no hayan datos faltantes y se calcula el promedio de estos valores, para obtener la acción de control final.
7. Esta acción de control es un valor entre $[0, 1]$ y se debe multiplicar por el tamaño del intervalo de decisión (15 minutos), para así obtener el tiempo durante el cual estará encendido el actuador respectivo en el siguiente intervalo. Posterior a esto se reinicia el proceso en el siguiente bloque de 15 minutos.

4.3. Arquitectura del controlador

El sistema se compone de cuatro sub-controladores; uno por cada variable (HS, TS, HA, TA) con su respectivo actuador (riego para variables de sustrato, y ventilación para variables ambientales). Este proceso de separación sirve para establecer ponderaciones (o ganancias) según qué tan importante es una variable por sobre otra para la acción de control del actuador respectivo, o qué tan alejado esté el resultado de control del esperado.

En la Figura 4.6 están representados estos sub-controladores con diagramas de bloques, un tipo de definición formal del área de control utilizada para detallar el proceso de cálculo interno que ocurre al ingresar una nueva entrada al controlador, calculada a partir de las variables medidas por los sensores y sus referencias.

Por un lado tenemos el controlador de riego, que tiene como variables de entrada la humedad de sustrato (HS) y la temperatura de sustrato (TS), además de sus valores de referencia (HSref y TSref). Con estos valores se calcula el error y la variación del error, para luego procesarlos con el sistema de lógica difusa de Mamdani diseñado en específico (sus conjuntos y reglas difusas) para cada una de estas variables. Primero se realiza la fuzzificación (transformación a variables difusas) de estas entradas, luego se determina la acción resultante (en base a sus reglas) y finalmente se transforma este resultado difuso a un valor determinístico. El resultado es ponderado por sus ganancias respectivas (gHS y gTs) para así obtener el

valor de riego total, que se aplicará al proceso para después comenzar un nuevo intervalo de decisión.

Este proceso es equivalente para el controlador de ventilación, cambiando simplemente las variables involucradas a la humedad ambiente (HA) y temperatura ambiente (TA), junto a sus respectivas referencias (H_{Aref} y T_{Aref}) y ganancias (g_{HA} y g_{TA}).

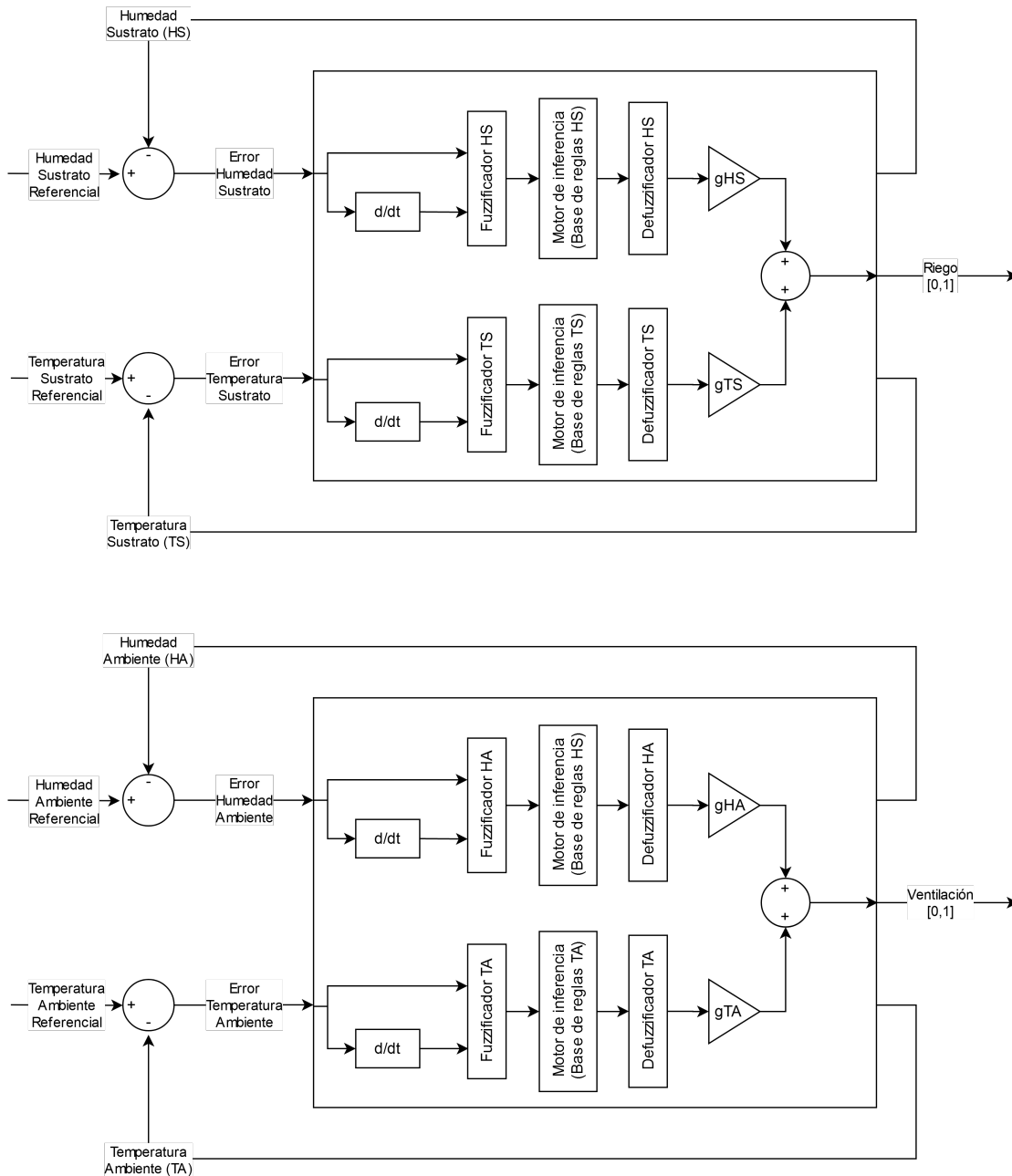


Figura 4.6: Controlador de riego (superior) y ventilación (inferior)

El diseño de los sistemas de lógica difusa tipo Mamdani propuestos para los sub-controladores requiere, como se indica previamente, el diseño de conjuntos difusos que se ajusten a cada variable considerada. En la siguiente sección se introduce una manera de generar estos conjuntos a partir de definiciones lingüísticas de conceptos relacionados al cultivo, pero que son de fácil comprensión para todo tipo de usuarios.

4.4. Conjuntos Difusos

En controladores del tipo error/delta se utilizan generalmente 5 conjuntos difusos: Negative Large, Negative, Zero, Positive, Positive Large, con funciones de pertenencia asociadas a una tolerancia del error y del incremento del error (delta) en las variables sensadas, que se definen a partir de opiniones de expertos en sistemas de este tipo. Sin embargo, en este proyecto se busca diseñar un prototipo lo más personalizable posible, por lo que se definirá la tolerancia (del error) a partir de un único atributo de la clase.

Esto se consigue a través de la definición de los siguientes rangos para los conjuntos difusos, que pueden amoldarse a los valores ideales o críticos de cualquier especie en función de los conceptos explorados en la Sección 2.1, aunque también puede modelarse a través del asesoramiento de expertos agrónomos u otro tipo. Estos conjuntos pretenden abarcar las clasificaciones que se hacen tradicionalmente a las variables de un cultivo:

1. **Mínimo / Máximo del Universo (MinU / MaxU)**: Es el rango correspondiente al universo dentro del cual la variable se moverá. Puede extraerse desde las especificaciones de los sensores o de la definición de rangos realistas para nuestra especie, ambiente y sensor. Lo importante es que el valor medido no salga nunca de este rango.
2. **Mínimo / Máximo Letal (MinL / MaxL)**: Es el rango tras el cual no hay posibilidad de supervivencia de la planta, o tras el cual se sufre un estrés/shock irreparable.
3. **Mínimo / Máximo Biológico (MinB / MaxB)**: Es el rango tras el cual una planta deja de producir su hoja, fruto o flor de forma normal, dado que las condiciones ambientales no son las recomendadas para la actividad biológica.
4. **Mínimo / Máximo Óptimo (MinO / MaxO)**: Es el rango tras el cual una planta no produce de forma óptima su hoja, fruto o flor. Puede considerarse igualmente algún aspecto en particular (dulzor, tamaño, etc.) si es clave para la especie. Dentro de este rango se obtienen los mejores resultados, y es dependiente de la etapa de crecimiento y el horario (día o noche).
5. **Mínimo / Máximo Específico (MinE / MaxE)**: Es el rango utilizado para forzar una regulación específica de las variables. Se puede ocupar para control de plagas, hongos o para especificar un ambiente más tropical o frío, dependiendo de la planta y la situación particular del entorno de cultivo.

Realizando este análisis para una planta de albahaca (*Ocimum basilicum*), que es representativa de la mayoría de las herbáceas y en particular es utilizada para las pruebas del sistema, se definen los siguientes valores para cada rango de las variables, presentados en la Tabla 4.1.

Variable	Germinación	Vegetación	Floración	Pre-Cosecha
Temperatura Sustrato (TS)	20-26°C (Día) 15-21°C (Noche)	20-28°C (Día) 15-23°C (Noche)	18-26°C (Día) 13-21°C (Noche)	18-26°C (Día) 13-21°C (Noche)
Humedad Sustrato (HS)	70-75 % (Día) 60-65 % (Noche)	55-65 % (Día) 45-55 % (Noche)	55-65 % (Día) 45-55 % (Noche)	55-65 % (Día) 45-55 % (Noche)
Temperatura Ambiental (TA)	20-25°C (Día) 15-20°C (Noche)	22-28°C (Día) 18-22°C (Noche)	20-26°C (Día) 15-21°C (Noche)	18-24°C (Día) 13-19°C (Noche)
Humedad Ambiental (HA)	65-80 % (Día) 40-50 % (Noche)	40-70 % (Día) 40-50 % (Noche)	40-50 % (Día) 40-50 % (Noche)	30-40 % (Día) 30-40 % (Noche)

Tabla 4.1: Rangos óptimos para cada variable

Para simplificar el diseño del programa, se limita la acción del sistema a la etapa de crecimiento vegetativo y durante el día. Para posibles extensiones del proyecto podrían incluirse al modelo los distintos ciclos de la planta y etapas del día. Con estas restricciones y con los resultados encontrados en la literatura, se establecen los siguientes valores para los rangos expuestos presentados en la Tabla 4.2

	MinU	MinL	MinB	MinO	MinE	Ref	MaxE	MaxO	MaxB	MaxL	MaxU
TS	-10	10	15	20	22	24	26	28	30	60	85
HS	0	20	30	55	58	60	62	65	80	90	100
TA	0	10	15	22	24	25	26	28	30	45	50
HA	20	25	30	40	50	55	60	70	80	85	90

Tabla 4.2: Rangos de las funciones de pertenencia para cada variable

A partir de estos valores y el valor de referencia indicado en azul, se utiliza la fórmula 4.1 para obtener los valores de cada rango del error. Este método permite relacionar directamente los valores ingresados por el usuario con los elementos que componen al controlador difuso, obteniendo una forma directa de personalizar el sistema a la necesidad y experiencia de cada usuario. Con esta transformación se obtiene:

Error	MinU	MinL	MinB	MinO	MinE	MaxE	MaxO	MaxB	MaxL	MaxU
TS	-61	-36	-6	-4	-2	2	4	9	14	34
HS	-40	-30	-20	-5	-2	2	5	30	40	60
TA	-25	-20	-5	-3	-1	1	3	10	15	25
HA	-35	-30	-25	-15	-5	5	15	25	30	35

Tabla 4.3: Rangos de las funciones de pertenencia del error de cada variable

El delta del error se determina a partir de una tolerancia establecida para la desviación de éste. El valor seleccionado proviene del análisis de la variación de los valores a lo largo del tiempo, observando cuánto cambian cuando por ejemplo, no existen acciones ejercidas sobre la variable, o el caso contrario en que un actuador modifica sus valores. Este número es ingresado como parámetro para el controlador (utilizando 1 para esta memoria), y se utiliza

como paso para la separación entre las cotas de cada rango. La excepción son los valores de los extremos que representan el universo del delta, equivalentes al máximo cambio posible para el error, como se muestra en la Tabla 4.4.

Delta	MinU	MinL	MinB	MinO	MinE	MaxE	MaxO	MaxB	MaxL	MaxU
TS	-95	-4	-3	-2	-1	1	2	3	4	95
TS	-100	-4	-3	-2	-1	1	2	3	4	100
TS	-50	-4	-3	-2	-1	1	2	3	4	50
TS	-70	-4	-3	-2	-1	1	2	3	4	70

Tabla 4.4: Rangos de las funciones de pertenencia del delta (del error) de cada variable

Con los valores de ambas tablas se definen los conjuntos difusos en base al esquema de la Figura 4.7. Cada punto representa un vértice de estas funciones de pertenencia trapezoidales.

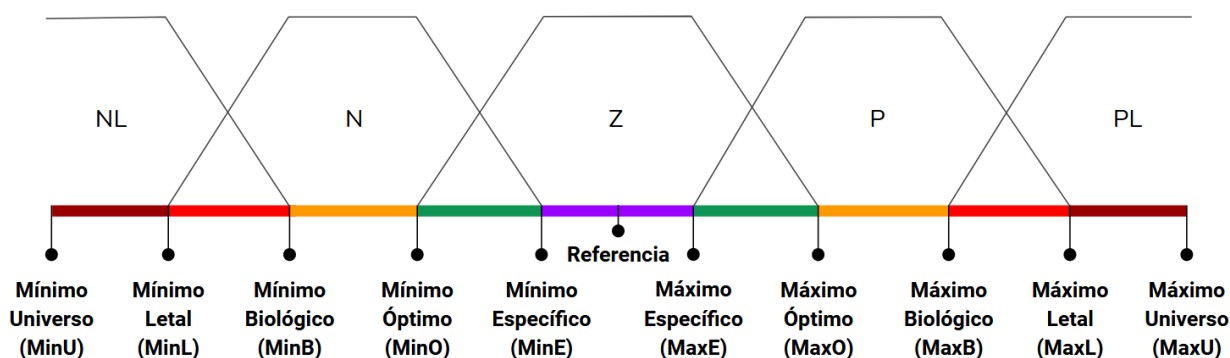


Figura 4.7: Funciones de pertenencia para los conjuntos de error y su incremento (delta)

Los resultados de estas transformaciones geométricas para el error y su incremento (delta) de cada variable son presentadas en las Figuras 4.8 a la 4.11. Para diseñar los conjuntos difusos de los actuadores, se realiza simplemente una distribución equitativa del universo $[0,1]$ en 4 subconjuntos: Zero, Low, Medium y High. Este universo se utiliza para que las acciones de control se correspondan con un porcentaje del intervalo de decisión. Se utiliza la función *automf* de Scikit Fuzzy para generar automáticamente estas funciones de pertenencia de tipo triangular para la cantidad de conjuntos que se asignen, tal como se observa en la Figura 4.12. Se propone como extensión a esta memoria, la realización de distribuciones más complejas, como por ejemplo, cargadas hacia el 0 o el 1, reduciendo así los subconjuntos y acercando las funciones a los bordes, pero eso dependerá del tipo de planta y por ahora se utiliza esta distribución.

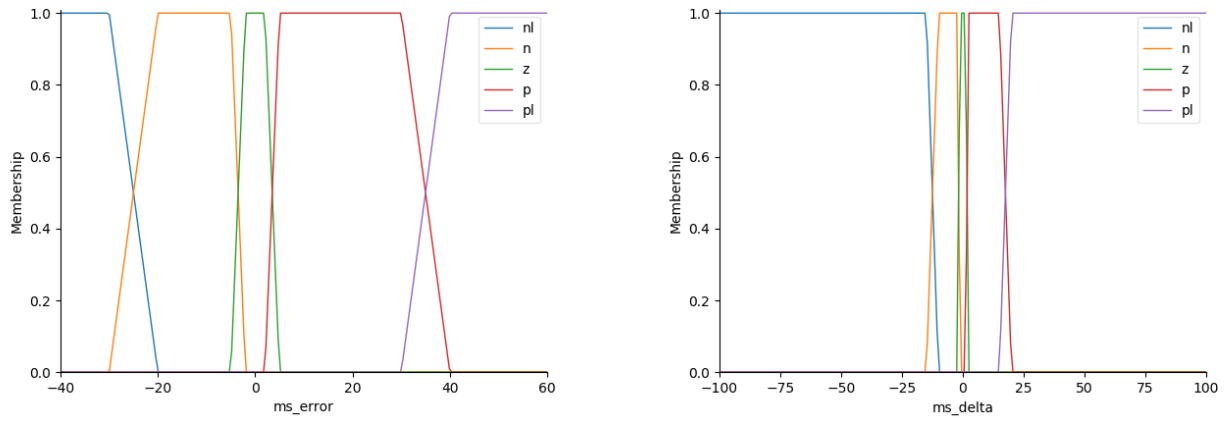


Figura 4.8: Funciones de pertenencia para el error y su incremento (humedad de sustrato)

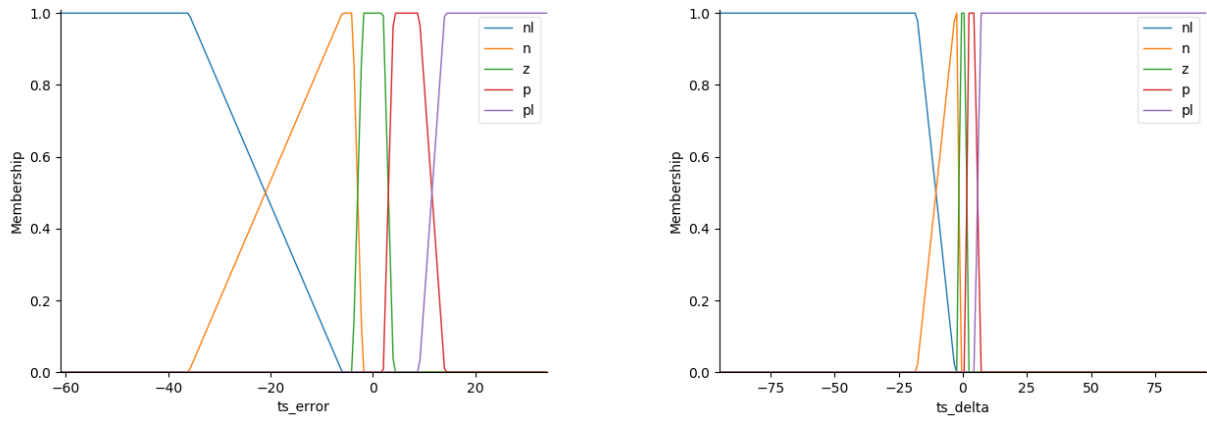


Figura 4.9: Funciones de pertenencia para el error y su incremento (temperatura de sustrato)

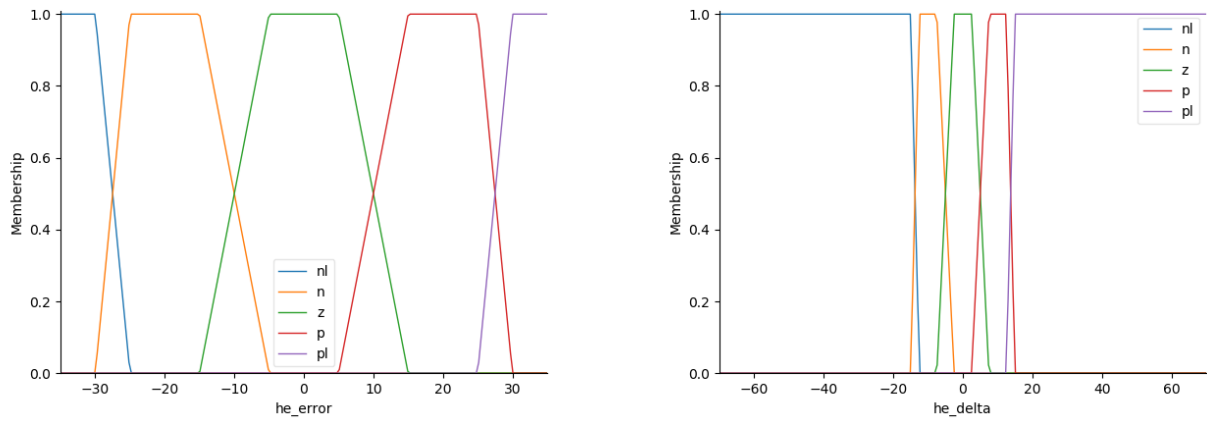


Figura 4.10: Funciones de pertenencia para el error y su incremento (humedad ambiente)

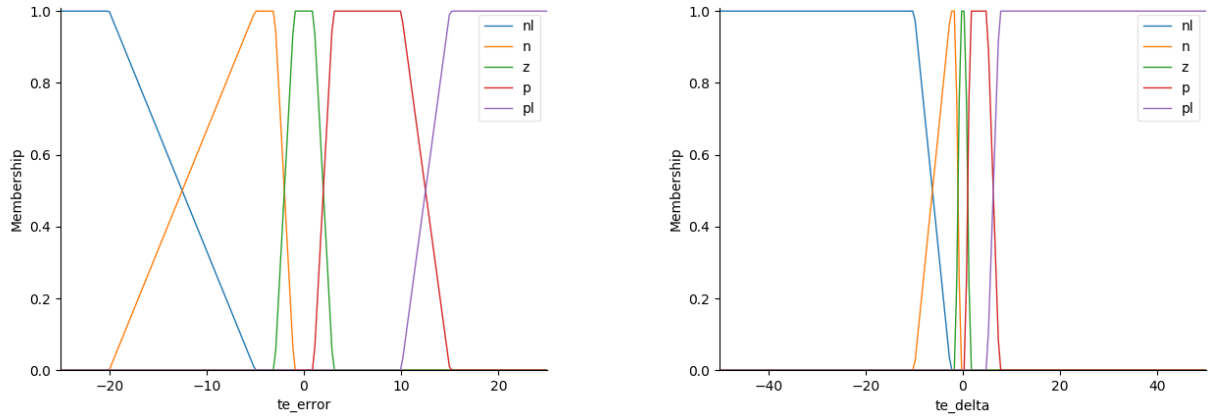


Figura 4.11: Funciones de pertenencia para el error y su incremento (temperatura ambiente)

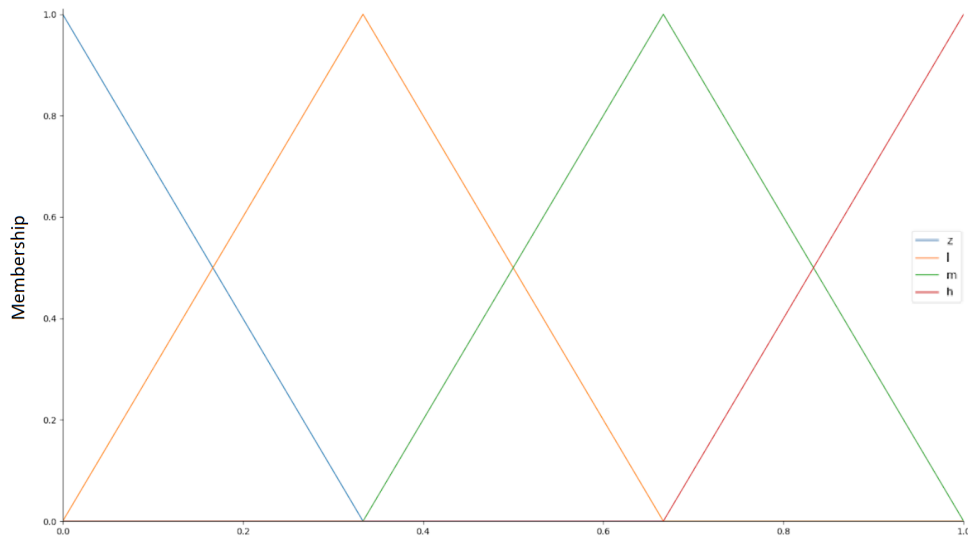


Figura 4.12: Función de pertenencia de los actuadores (riego y ventilación)

4.5. Reglas Difusas

El conjunto de reglas difusas son la base del accionar del controlador. A partir de los conjuntos difusos del error y su incremento (delta) de cada variable, tendremos una serie de combinaciones generadas por cada par ($error$, $\Delta error$) de funciones de pertenencia activadas en base a los valores obtenidos en los sensores. Cada combinación tiene como consecuencia la activación de una función de pertenencia del actuador correspondiente. A este conjunto de elementos se les llama reglas difusas, y buscan reflejar la acción que se toma en cada escenario del sistema, como por ejemplo cuánto se debe regar si el sustrato está muy seco ($error: PL$) y si además se seca con mucha rapidez ($\Delta error: PL$). Esta acción dependerá de la influencia que tiene el actuador sobre nuestra variable, observando primero si es proporcional o inversa a su crecimiento, y luego el impacto (magnitud) que tiene sobre ésta. En la Tabla 4.5 se presentan estas reglas para cada par ($error$, $\Delta error$) de las variable de interés, obteniendo como resultado una variable difusa para el actuador respectivo.

Error \ Delta	NL	N	Z	P	PL
NL	Z	Z	Z	Z	Z
N	Z	Z	Z	Z	Z
Z	Z	Z	Z	Z	L
P	Z	Z	L	M	H
PL	Z	L	M	H	H

Tabla 4.5: Reglas difusas de la humedad de sustrato y el riego

Error \ Delta	NL	N	Z	P	PL
NL	H	M	L	Z	Z
N	H	M	Z	Z	Z
Z	Z	Z	Z	Z	Z
P	Z	Z	Z	Z	Z
PL	Z	Z	Z	Z	Z

Tabla 4.6: Reglas difusas de la temperatura de sustrato y el riego

Error \ Delta	NL	N	Z	P	PL
NL	H	M	M	L	Z
N	H	M	L	Z	Z
Z	L	L	Z	Z	Z
P	Z	Z	Z	Z	Z
PL	Z	Z	Z	Z	Z

Tabla 4.7: Reglas difusas de la humedad ambiente y la ventilación

Error \ Delta	NL	N	Z	P	PL
NL	H	M	M	L	Z
N	M	M	L	Z	Z
Z	L	Z	Z	Z	Z
P	Z	Z	Z	Z	Z
PL	Z	Z	Z	Z	Z

Tabla 4.8: Reglas difusas de la temperatura ambiente y la ventilación

Estos valores podrían ser escogidos por el usuario, pero requiere de conocimientos y experiencia previa en la dinámica de las variables y actuadores seleccionados. En este caso, dichos valores se especifican directamente en base a pruebas realizadas en distintos escenarios, y a la comparación de los resultados obtenidos con el comportamiento esperado del sistema.

En la Figura 4.13 se presenta un ejemplo del funcionamiento del mecanismo de activación

de las reglas difusas para el riego, cuando la humedad del sustrato es 46 % en dos mediciones seguidas (o sea, el delta del error es 0).

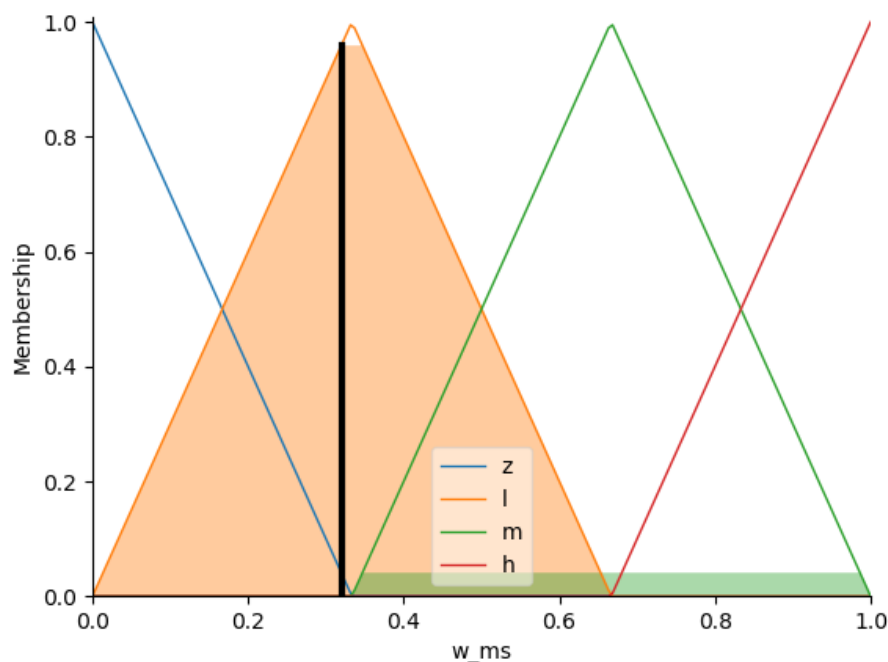


Figura 4.13: Ejemplo de activación del riego según la humedad de sustrato

Como se observa, los conjuntos activados (coloreados) son L y M para el riego. Sin embargo, el par ($error$, $\Delta error$) es mayormente Low que Medium. Para transformar esta noción a un valor porcentual que pueda utilizarse en los actuadores, se necesita un método de defusificación. De las alternativas presentadas en la Figura 4.3, se selecciona el método *minimum of maximum* que obtiene el valor mínimo de los puntos con máximo valor de activación. En este caso eso equivale a una acción resultante cercana a 0,3 de riego (línea negra), dado que los puntos con máximo valor de activación son los del conjunto L.

El beneficio de este método es que nos permite minimizar el uso de los actuadores, ya que cuando se activan conjuntos cercanos al cero (máximo valor de activación en el Zero), al tomar el menor de los máximos tendremos una acción nula, y así se reducirán los porcentajes pequeños de acción y por ende la sobrecarga del actuador. La consecuencia negativa de este método es que reduce el resto de acciones, haciendo que la dinámica de cambio de los valores sea más lenta. Para cultivos urbanos y periurbanos esto no representa mayor inconveniente, ya que no se ven afectados de forma crítica por pequeños retrasos en el proceso de control.

4.6. Algoritmo de control implementado

Como se detalla en la Figura 3.7, el controlador es una instancia creada dentro del data collector. Este último le entrega los datos recolectados en el sistema, para así determinar las acciones de control correspondientes. Los pasos presentados en el Algoritmo 1, son ejecutados

consecutivamente al inicializar el controlador por medio de su constructor. Se omiten algunos atributos en función de mostrar solamente las funciones y elementos más representativos de la dinámica del sistema. Al final de este procedimiento tendremos las cuatro simulaciones listas para recibir los datos de cada variable.

Algorithm 1 Inicialización del controlador difuso

- 1: **procedure** CONTROLLER(time_interval, references, ranges, gains, tolerance)
 - 2: **initUniversesAndFuzzyVariables**() ▷ Transforma los rangos de caracterización y las referencias a los conjuntos difusos de error y delta
 - 3: **initMembershipFunctions**() ▷ Genera las funciones de pertenencia para la entrada a partir de los conjuntos creados
 - 4: **initRules**() ▷ Transforma las tablas de reglas propuestas al formato requerido por Scikit-Fuzzy
 - 5: **initControllers**() ▷ Genera un controlador por cada variable a partir de las reglas generadas
 - 6: **initSimulations**() ▷ Genera una instancia de cada controlador para procesar los datos ingresados
-

Al final de cada intervalo de medición (1 minuto), el gateway envía el valor promedio de la variable sensada al data collector, que posteriormente lo retransmite al controlador difuso para ejecutar el bloque de funciones del Algoritmo 2. Sin embargo, el retorno de la acción de control solamente ocurre cuando se ha completado el intervalo de decisión, luego del cual se escriben las acciones de control del actuador en su tópico correspondiente. Al final de ese proceso se realiza la limpieza de las acciones de control, para así reiniciar el proceso en el siguiente intervalo asegurando la disponibilidad de la memoria requerida.

Algorithm 2 Procesamiento de datos con controlador difuso

- 1: **procedure** CONTROLLERPROCESSING(data={measurement,value})
 - 2: **saveData**(data) ▷ Guarda el nuevo dato en un buffer temporal
 - 3: **calculateErrorandDelta**(measurement) ▷ Calcula el error y el delta de la variable, y los guarda en buffers temporales
 - 4: **process**(measurement)* ▷ Al final del intervalo de decisión, calcula el promedio de las acciones de control con los datos guardados en los buffers temporales de la variable, y lo guarda
 - 5: **calculateOutput**(measurement)* ▷ Pondera el promedio de cada variable por su ganancia, y retorna la acción de control final cuando todas las dependencias del actuador tienen su valor calculado
 - 6: **cleanOutput**(measurement) ▷ Realiza la limpieza de las acciones de control guardadas
-

Casi todas las funciones son simples y realizan las acciones/operaciones indicadas en los comentarios descritos en el algoritmo. Las funciones marcadas con un asterisco se profundizan en el Algoritmo 3 para aclarar el flujo de datos del sistema.

Algorithm 3 Funciones del controlador difuso

```

1: procedure PROCESS(measurement)
2:   if size(data[measurement]) % measure_interval is 0 then:
3:     for error, delta in buffers do:
4:       computeOutput(measurement, error, delta)*
5:       output[measurement] = mean(outputs[measurement])

6: procedure COMPUTEOUTPUT(measurement, error, delta)
7:   sim = simulations[measurement]
8:   sim.input('error') = error
9:   sim.input('delta') = delta
10:  sim.compute()
11:  output = sim.output[measurement]
12:  outputs[measurement].add(output)

13: procedure CALCULATEOUTPUT(measurement)
14:  result[measurement] += output[measurement] * gain[measurement]
15:  if waiting[measurement] then:
16:    return result
17:  waiting[measurement] = not(waiting[measurement])

```

4.7. Discusión

En este capítulo se describió en detalle el sistema de control difuso incluyendo los controladores, las variables consideradas (con sus respectivos rangos), los conjuntos y reglas difusas, y los algoritmos de control implementados. En el siguiente capítulo se describe la evaluación del comportamiento del sistema, el cual está implementado sobre la infraestructura descrita en el Capítulo 3.

Capítulo 5

Evaluación de la Solución Propuesta

En el presente capítulo se exploran los resultados e impacto de la solución propuesta. Para asegurar un funcionamiento óptimo y acorde a los objetivos planteados, se entrega una serie de pruebas que permiten demostrar el éxito y fiabilidad del sistema, respaldando estas afirmaciones con información empírica del control de las variables seleccionadas.

5.1. Escenario de Pruebas

Se utiliza para las pruebas una carpa de cultivo de interior Cropbox de 60x60x120cm, dotada de una luz LED COB 3500K 100W y dos albahacas tradicionales (*Ocimum basilicum*) en maceteros de 5L. En el lugar se coloca igualmente el sistema electrónico, junto al estanque de agua requerido para el riego automatizado. La arquitectura propuesta en la Figura 1.1 es instanciada con los sensores y controladores seleccionados en la Sección 2.9, resultando el esquema de la Figura 5.1, enumerando todos los componentes involucrados.

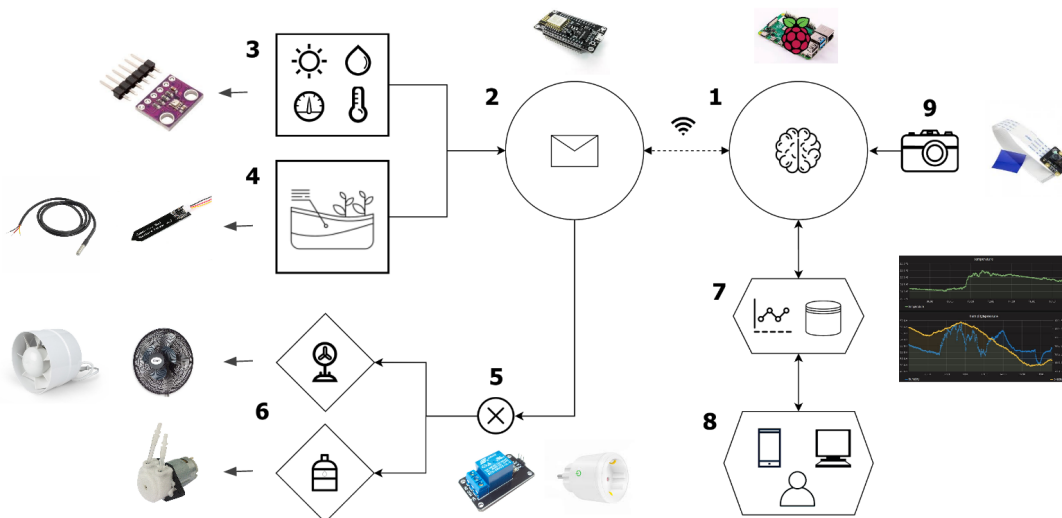


Figura 5.1: Instancia de la arquitectura física

1. **Raspberry Pi 3B+**
2. **NodeMCU ESP8266 v3**
3. Sensor de humedad y temperatura ambiente **BME280**
4. Sensor de temperatura de sustrato **DS18B20** y sensor de humedad de sustrato **Capacitive Soil Moisture Sensor v1.2**
5. **Relé 1 Canal 220V y WiFi Smart Plug Smart Life**
6. **Sistema de ventilación y sistema de riego**
7. Base de Datos **InfluxDB** y framework para visualizaciones **Grafana**
8. Usuario
9. Cámara nocturna **Pi NoIR**

A lo largo de las pruebas de funcionamiento y análisis preliminares, se observó un problema crucial en el sistema. Los valores de humedad ambiental registrados a lo largo del día eran demasiado bajos, manteniéndose en su mayoría entre el rango biológico y letal (25 % a 30 %), provocando que el sistema de ventilación nunca fuera activado para corregir esta variable. Para solucionar este defecto se incluye un humidificador de 6L Cornwall Electronics al escenario, el cual provee este porcentaje extra de humedad requerida. La incorporación tardía de este equipo evitó que fuese agregado como actuador. Sin embargo, se propone su implementación en el futuro, dado que el agua desmineralizada que utiliza este dispositivo se agota rápidamente, por lo que un sistema óptimo requiere también utilizar eficientemente este recurso.

Los equipos externos al sistema se configuran en base a horarios de activación, utilizando para esto, dos timer digitales enchufables Cornwall Electronics. Estos timers cuentan con alternativas diarias o semanales para ejecutar los cambios de estado. Las configuraciones escogidas fueron las siguientes:

1. **Luz:** 06:00 a 24:00 (18 horas de luz)
2. **Humidificador:** 10:00 - 12:00, 14:00 - 16:00, 18:00 - 20:00, 22:00 - 24:00

Para visualizar las métricas, notificaciones y alertas se utiliza el software Grafana, que entrega además una serie de plugins e interfaces para realizar consultas a la base de datos a través del lenguaje InfluxQL, y así simplificar los análisis propuestos. Del sistema base se utilizan primero, los gráficos de línea para mostrar la información de los actuadores y métricas, agregando en estas últimas los límites de sus rangos de caracterización con los colores previamente definidos en la Figura 4.7, y segundo, las alertas que nos permiten definir una condición que gatilla acciones dentro de Grafana. En este caso, la acción es el envío de notificaciones al dispositivo móvil a través del bot de Telegram (BotFather). Este bot informa tanto las acciones ejecutadas (encendido y apagado), como las situaciones críticas de cada variable cuando su valor está fuera del rango biológico por al menos 5 minutos.

Además de estos componentes, se agrega un plugin llamado *Video Panel by Innuius* para la visualización en tiempo real de nuestro sistema. Esto se implementó usando un iframe (HTML) que transmite en formato streaming las imágenes obtenidas de la cámara, a partir de un código escrito en Python. El resultado se muestra en la Figura 5.2 que contempla los elementos descritos mostrando los datos de las últimas 24 horas. A continuación se describen las pruebas realizadas al sistema.

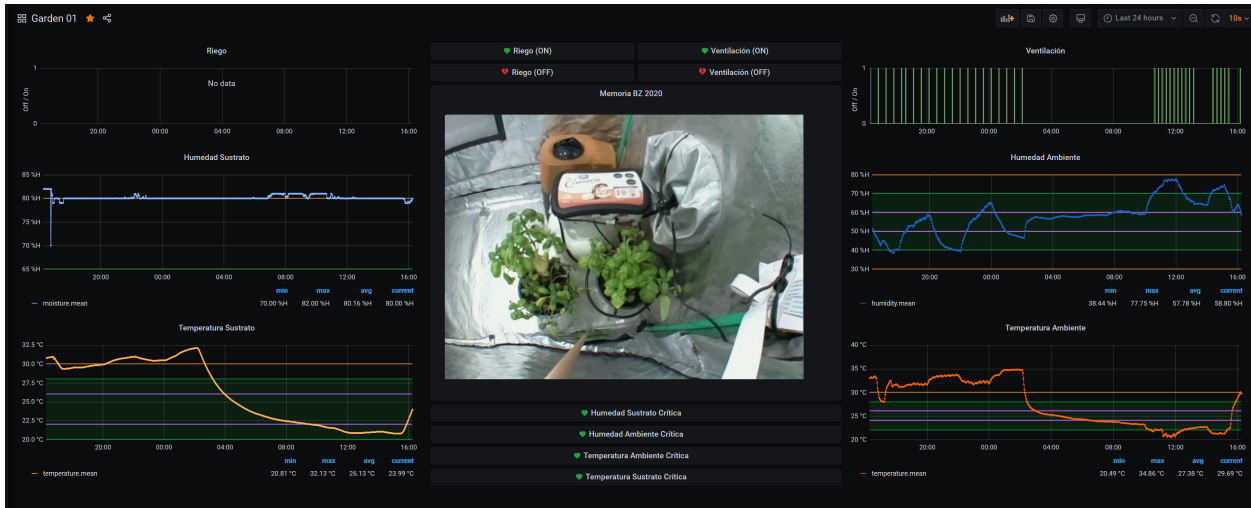


Figura 5.2: Interfaz gráfica con información de las últimas 24 hrs.

Con este escenario montado y los componentes de hardware y software funcionando, podremos evaluar los resultados del control a partir de las pruebas presentadas en la siguiente sección.

5.2. Pruebas realizadas

Las pruebas seleccionadas buscan cubrir el mayor espectro de situaciones en el escenario de control. Para esto se analizan las siguientes situaciones: cambio de las variables en un lapso pequeño de tiempo (Sección 5.2.1), análisis del control a corto plazo de variables directamente afectadas por nuestros actuadores (Sección 5.2.2), análisis del control a largo plazo de todas las variables del sistema (Sección 5.2.3), comparación de este último proceso de control con el trabajo manual (Sección 5.2.4) y finalmente la revisión de alertas y notificaciones recibidas en el dispositivo móvil en relación a los datos reales del sistema (Sección 5.2.5).

5.2.1. Funcionamiento básico del controlador difuso

La primera prueba buscó comprobar el correcto funcionamiento del sistema, comparando las acciones planteadas en las reglas difusas (Sección 4.5), con las acciones realmente ejecutadas tras un lapso de 15 minuto (intervalo de decisión). El método utilizado para explorar la mayor gama de respuestas corresponde a la alteración de las variables de forma artificial, utilizando para esto equipos y/o estímulos externos que permitieron situar al sistema entre varios estados de forma rápida, realizando por ejemplo cambios a la temperatura de sustrato con agua a diferentes grados, o alteraciones a la humedad de sustrato utilizando varias muestras con distintas concentraciones de agua. A través de la ejecución de estas pruebas

se encontró un problema con la baja velocidad de cambio de las variables ambientales, y para solucionarlo, se implementó un código que simula el crecimiento/decrecimiento de las variables en un lapso fijo de tiempo.

En la Tabla 5.1 se presentan los distintos escenarios cuya transición (de estado inicial a final) se limita al intervalo de decisión (15 min.), esperando detectar la mayor cantidad de puntos representativos del rango de acciones posibles, incluyendo cada una de las reglas de los actuadores (Zero, Low, Medium, High). Como las acciones de control están distribuidas uniformemente entre 0 y 1, son de fácil análisis numérico para esta prueba (Figura 4.12). Es importante destacar que para analizar cada variable por separado, se establecieron las ganancias de sus respectivas co-dependencias para el actuador (humedad a temperatura y viceversa) como 0.

ID	Variable	Estado inicial	Estado final	Acción esperada	Acción real [min]
1	Humedad Sustrato (HS)	Óptima	Alta	Zero	0
2	HS	Baja	Óptima	Low	4
3	HS	Alta	Baja	Medium	7
4	HS	Óptima	Baja	Medium o High	11
5	Temperatura Sustrato (TS)	Óptima	Baja	Zero	0
6	TS	Alta	Óptima	Medium	3
7	TS	Baja	Alta	Low	5
8	TS	Óptima	Alta	Medium o High	9
9	Humedad Ambiente (HA)	Óptima	Baja	Zero	0
10	HA	Alta	Óptima	Low	2
11	HA	Baja	Alta	Medium	2
12	HA	Óptima	Alta	Medium o High	4
13	Temperatura Ambiente (TA)	Óptima	Baja	Zero	0
14	TA	Alta	Óptima	Low	2
15	TA	Baja	Alta	Medium	7
16	TA	Óptima	Alta	Medium o High	10

Tabla 5.1: Casos de prueba y resultados de manipulación de variables

Como se observa en cada prueba, existe una transición entre más de un estado, y por ende, se definen las acciones esperadas como una ponderación del promedio de conjuntos activados. La activación individual de cada conjunto error/delta (sin variación) fue testeada en el desarrollo del código del controlador difuso, y es omitida para estas pruebas.

Los resultados de las pruebas se muestran en las Figuras 5.3 a la 5.6 a través de capturas de la interfaz gráfica. Allí se indican con anotaciones (funcionalidad incorporada en Grafana)

las dos etapas del proceso de control: 1) el intervalo de medición (15 min.) en el gráfico de la variable, y 2) el intervalo de ejecución de las acciones (0-15 min.) en el gráfico del actuador.



1) De óptima a alta



2) De baja a óptima

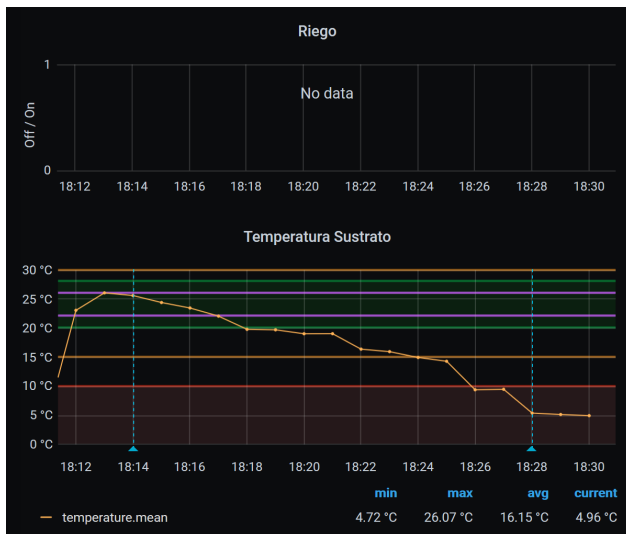


3) De alta a baja

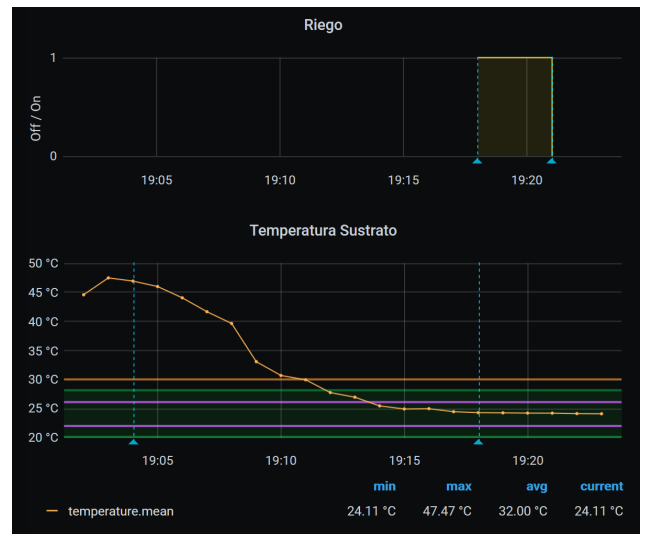


4) De óptima a baja

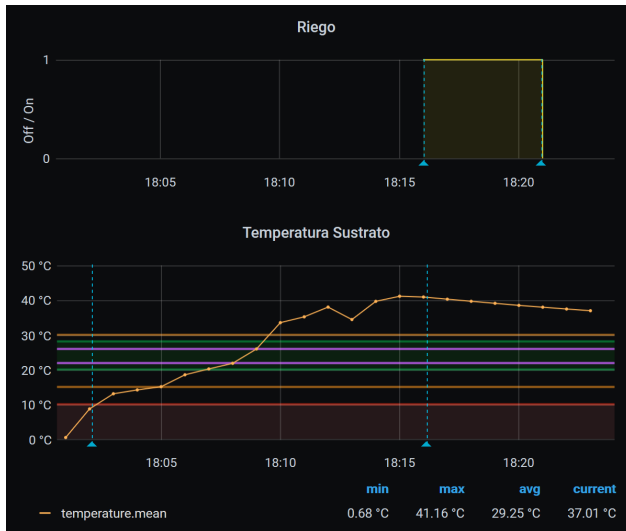
Figura 5.3: Resultados variación rápida de humedad de sustrato



5) De óptima a baja



6) De alta a óptima

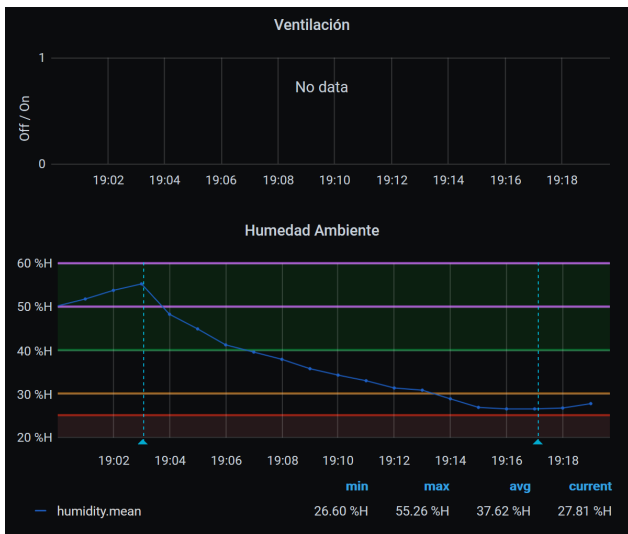


7) De baja a alta

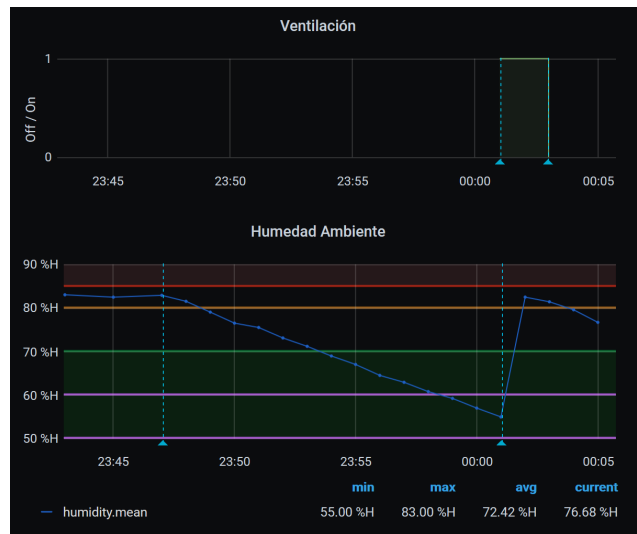


8) De óptima a alta

Figura 5.4: Resultados variación rápida de temperatura de sustrato



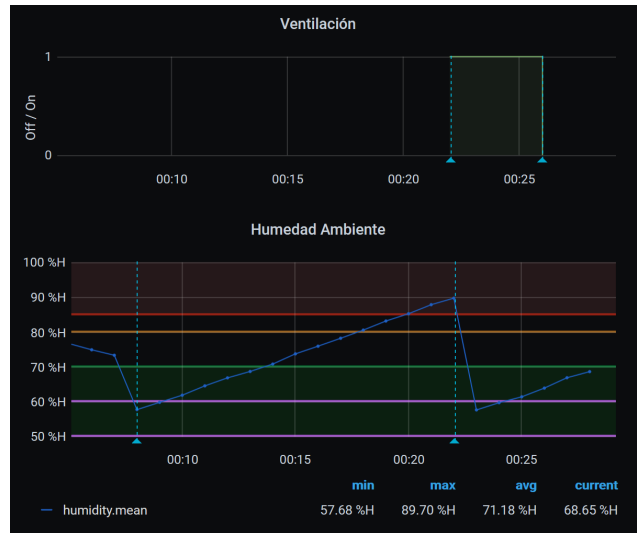
9) De óptima a baja



10) De alta a óptima

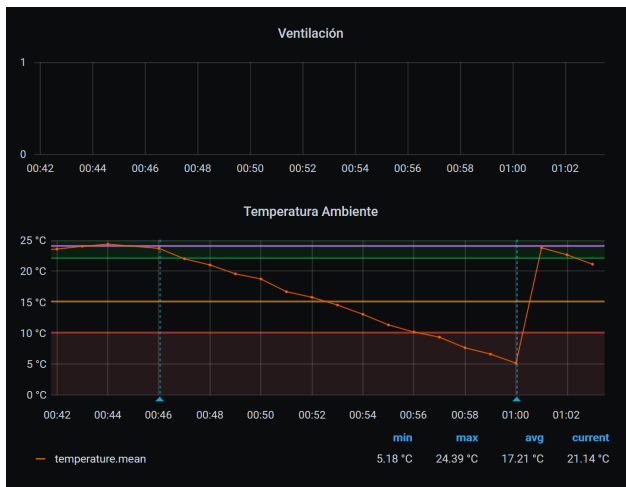


11) De baja a alta

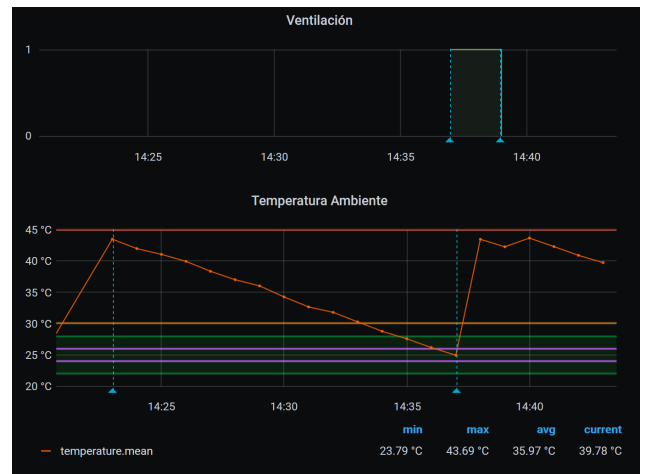


12) De óptima a alta

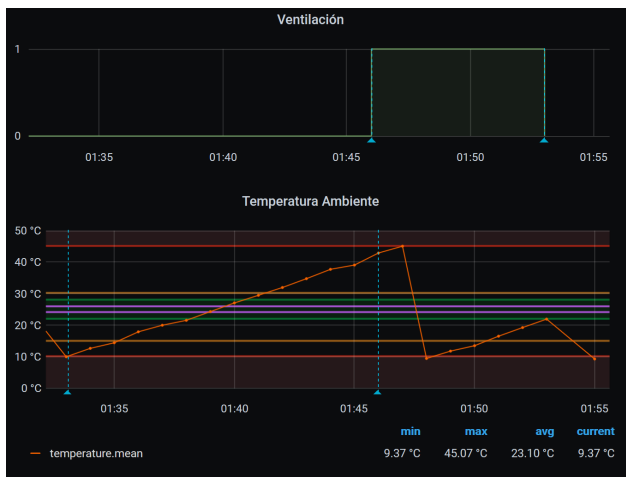
Figura 5.5: Resultados variación rápida de humedad ambiente



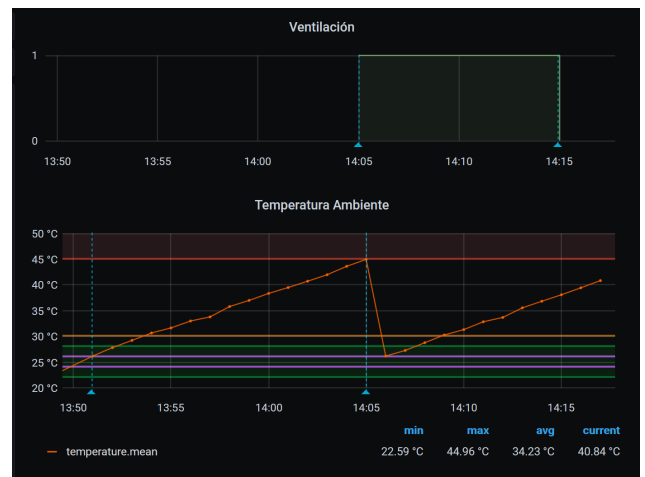
13) De óptima a baja



14) De alta a óptima



15) De alta a baja



16) De óptima a alta

Figura 5.6: Resultados variación rápida de temperatura ambiente

5.2.2. Control a corto plazo de humedad de sustrato y ambiente

La segunda prueba consistió en aislar las variables que son directamente alteradas por los actuadores seleccionados, para así ejecutar un análisis práctico sobre la precisión del control, y su impacto en la dinámica de las variables a lo largo de este proceso.

Para esto se seleccionan dos variables: 1) la humedad de sustrato (HS), ya que su valor es incrementado únicamente según la cantidad de agua (o tiempo) utilizada para el riego, y 2) la humedad de ambiente (HA), pues la única forma de contrarrestar el incremento de esta variable (por la acción del humidificador), es con la activación del sistema de ventilación y extracción.

Para este análisis se realiza la misma prueba en ambas variables. Primero, se desconectan los actuadores para inhibir las acciones de control, y se espera que el valor de cada variable esté al menos 15 minutos tras un punto crítico, cercano al rango biológico ($HS \leq 30\%$ y $HA \geq 75\%$). Transcurrido este lapso de tiempo, se re-conectan los actuadores antes del inicio de

la primera acción de control, y se observa el comportamiento de las mediciones en respuesta a las acciones ejecutadas. Se espera hasta alcanzar la estabilidad de las variables, para luego realizar consultas (InfluxQL) y calcular algunas métricas de desempeño.

Los resultados de ambas pruebas se muestran en las Figuras 5.7 y 5.8 a través de capturas de la interfaz gráfica, marcando con anotaciones cada una de las etapas descritas (agregando el tiempo de estabilización explicado más adelante).

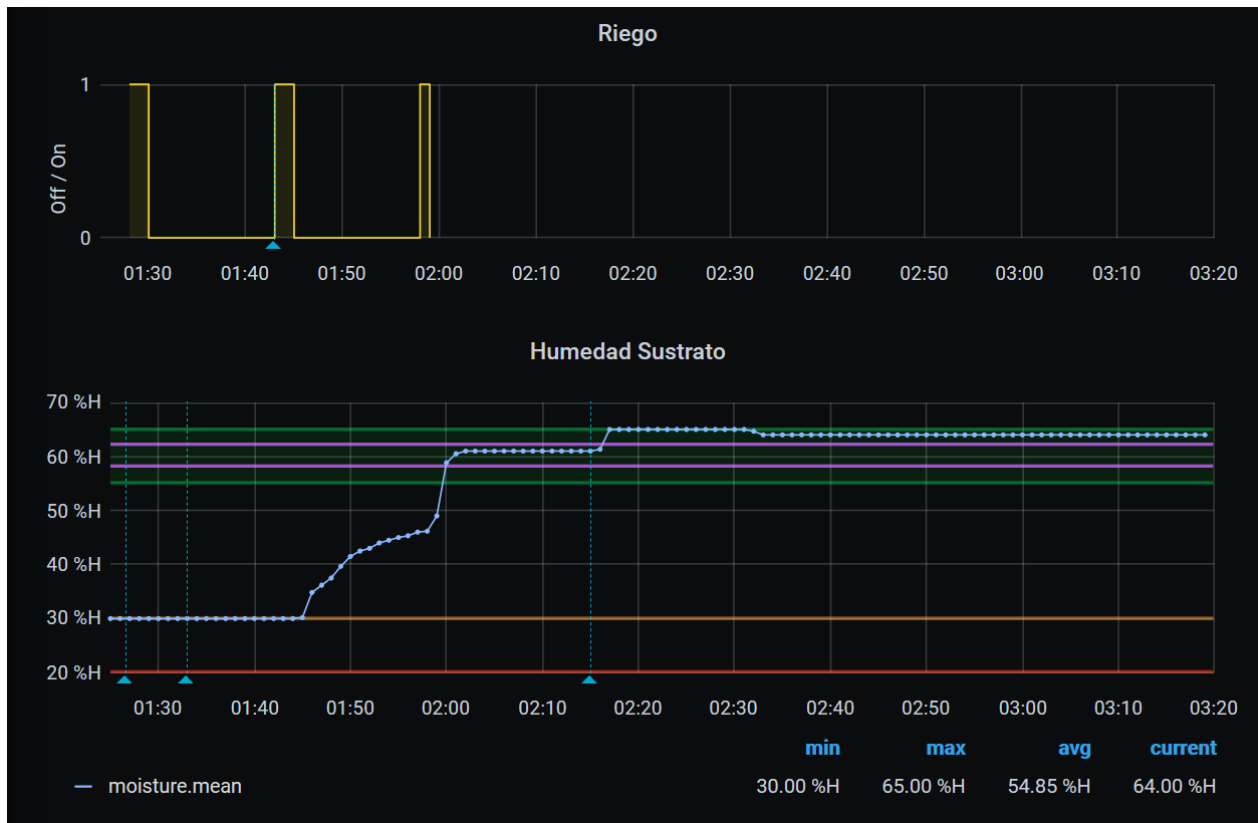


Figura 5.7: Resultados de control de Humedad de Sustrato

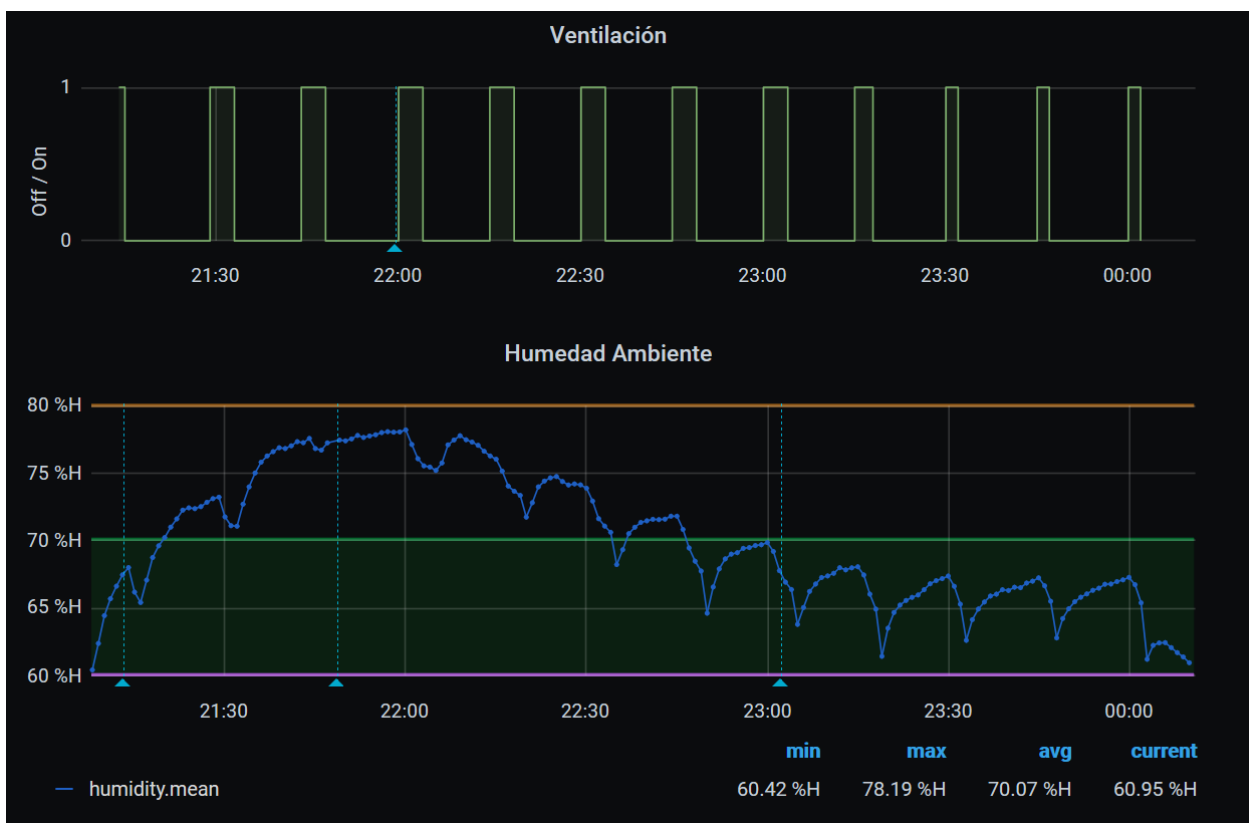


Figura 5.8: Resultados de control de Humedad de Ambiente

En la Tabla 5.2 se entregan los resultados numéricos de estas pruebas. Es importante aclarar que las métricas de error MAE (Mean absolute error) y $RMSE$ (Root mean square error) son calculadas en un lapso de 1 hora después de transcurrido el TDE (Tiempo de estabilización), o sea, el tiempo necesario para que la variable se mantenga dentro del rango óptimo (al menos durante 15 minutos). El $Peak$ representa el valor máximo (o mínimo) alcanzado, y se muestra para indicar por cuánto se sobrepasa el valor de referencia una vez que comienzan las acciones de control.

Variables	MAE	RMSE	Peak	TDE (min.)
Humedad Sustrato	4.04 %	4.04 %	65(+5) %	32
Humedad Ambiente	10.79 %	10.67 %	61(-6) %	62

Tabla 5.2: Métricas de desempeño para el control a corto plazo

Los resultados del control de la humedad de sustrato son bastante buenos, a pesar de que se obtiene un exceso de 5 % al ejecutar las acciones de control, por lo que hay que considerar un posible overshoot a largo plazo (como se verá en la siguiente prueba). Los resultados del control de la humedad de ambiente se consideran no muy buenos. La principal razón es que es mucho más fácil ingresar humedad al ecosistema que removerla, como es el caso del control propuesto (sobre la ventilación y no sobre el humidificador). Este problema se ve reflejado en el largo tiempo de estabilización obtenido y los altos valores de las métricas. Sin embargo,

un punto destacable es que nunca se sobrepasa el valor de referencia establecido porque la magnitud de cambio también es bastante pequeña.

5.2.3. Control a largo plazo (10 días)

La tercera prueba entregó resultados sobre la autosuficiencia y precisión del sistema en un caso real. Para este análisis se observó el comportamiento de las variables y sus actuadores en un lapso de 10 días, dentro de las cuales no se realizó ningún tipo de modificaciones o interrupciones al sistema, ni a la planta. Lo único que se hizo fue remover hojas en mal estado y chequear que no existieran fugas de agua o fallas eléctricas.

En este periodo de tiempo se analizó el detalle del cambio de los valores bajo la acción automatizada de ambos actuadores, cuantificando estos resultados con las mismas métricas de error escogidas en las pruebas de la sección anterior (*MAE* y *RMSE*). A priori, se esperaba una variación mucho mayor, pues este lapso de tiempo tan extendido involucra un mayor número de factores externos de cambio (ambientales, lumínicos, etc.) que producen variaciones imprevistas para estas métricas. Por esta razón, se incluye igualmente una captura de la interfaz gráfica de estos 10 días en la Figura 5.9, para así comprender el contexto del escenario de control. Los resultados numéricos de las métricas son presentados en la Tabla 5.3.

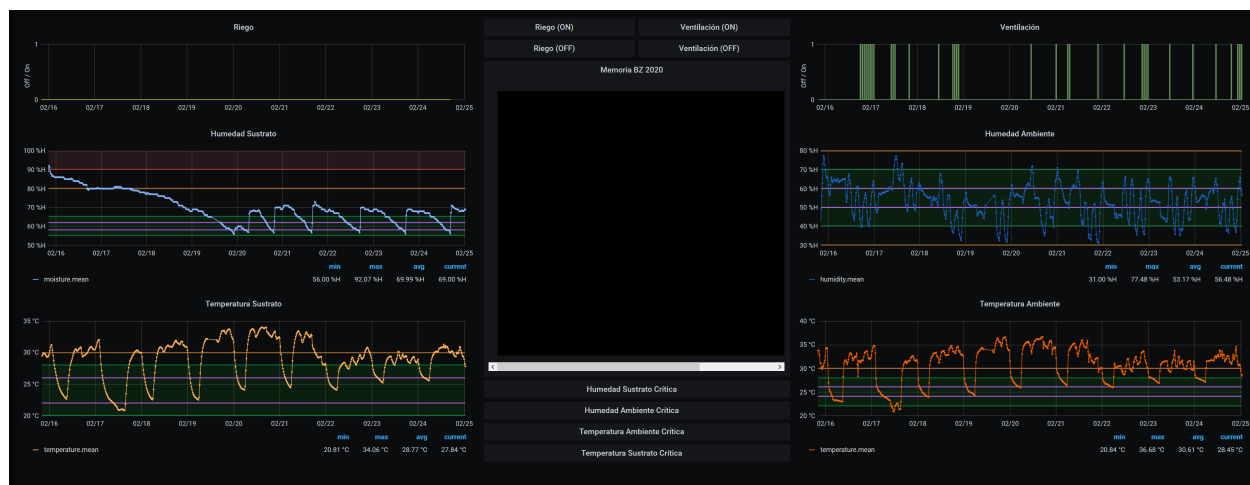


Figura 5.9: Resultados de la primera semana de control

El problema de las capturas con ventanas temporales tan extendidas, es que el software Grafana realiza una discretización que invisibiliza gran parte de las acciones de control, como se observa claramente en el gráfico de riego.

Contrario a lo esperado, los errores obtenidos son más bajos para la humedad ambiente, y más altos para la humedad de sustrato. Esto se explica ya que al comenzar las pruebas con el sustrato regado de forma manual, ocurrió un exceso de humedad involuntario, el cual tiene un descenso bastante lento, aumentando el valor del error en gran medida. Por otro lado, el error en la humedad ambiente es menor, ya que no se esperó a que el nivel de humedad fuera muy alto para actuar (como en la prueba a corto plazo), si no que era constantemente regulada por el sistema de ventilación, lo que se traduce en esos valores. Igualmente se debe

Variabes	MAE	RMSE
Humedad Sustrato	8,58 %	9,22 %
Temperatura Sustrato	5,28 %	5,78 %
Humedad Ambiente	7,58 %	9,55 %
Temperatura Ambiente	6,07 %	6,7 %

Tabla 5.3: Métricas de desempeño para control a largo plazo

destacar que en este caso si se obtienen *Peak*'s mucho más elevados para ambas métricas, obteniendo para la humedad de sustrato un 73(+13) % y para la humedad de ambiente un 31(+24) %, aunque ocurrió en horarios donde el humidificador no estaba encendido.

De las otras dos variables (temperatura de sustrato y ambiente) se observa que la mayor variación en sus valores está sujeta principalmente al horario lumínico, por sobre la acción de los actuadores. Sin embargo, estas últimas apoyaron en parte este proceso de cambio, logrando bajos porcentajes de error en estos 10 días.

5.2.4. Comparación entre el control a largo plazo y el riego manual

En paralelo a la prueba de control a largo plazo se realizó el ejercicio de riego manual sobre la segunda planta de albahaca (disponible en el escenario de pruebas), dentro del mismo intervalo de 10 días. Esta prueba buscó evidenciar la optimización de tres aspectos clave:

1. **Tiempo de trabajo invertido:** Se obtiene a partir de la cuantificación del tiempo utilizado en las tareas de riego manual con un cronómetro (del dispositivo móvil).
2. **Cantidad de agua utilizada para riego:** Se calcula la medición del agua utilizada en el riego automatizado a partir de la diferencia entre la cantidad de litros inicial y final del estanque. El cálculo de la cantidad utilizada en el riego manual es directa al ocupar una jarra con medidas en ml.
3. **Apariencia y bienestar de la planta:** Se analiza a partir de la captura de fotografías con la cámara del sistema, para presentar imágenes que permitan evaluar el progreso de ambas plantas.

Medición	Día										
	1	2	3	4	5	6	7	8	9	10	Total
Tiempo de trabajo (min)	0.54	1,73	0,67	1,52	4,15	0,71	2,51	0,64	1,72	1,33	15,52
Riego manual (ml)	0	150	0	300	125	0	250	0	250	125	1200
Riego automatizado (ml)	7000	-	-	-	-	-	-	-	-	5000	2000

Tabla 5.4: Resultados de comparación con riego manual

Los resultados se presentan en la Tabla 5.4 y la Figura 5.10. Notar que cada día se invierte tiempo en revisar el estado del sustrato para tomar la decisión de regar y en qué cantidad.

Revisaremos los aspectos en el orden presentado. Primero, la reducción de tiempo de trabajo para estos 10 días, fue de un total de 15:30 minutos aproximadamente. Este valor es referencial, ya que la proximidad y fácil acceso al agua de nuestro escenario de pruebas reducen grandemente esta carga. A pesar de que no es una gran cantidad de tiempo, los resultados son bastante positivos, ya que al estar distribuida esta cantidad a lo largo de 10 días, significará para el usuario una independencia asegurada de su cultivo en este lapso, al menos con respecto al riego.

Segundo, vemos que contrario a la hipótesis planteada no hubo una reducción en la cantidad de agua utilizada, de hecho se ocupan 800ml más. Sin embargo, hay dos aspectos a considerar: primero, el riego ideal (o cercano a este) se acerca más a los valores ocupados en la versión automatizada (como se aprecia en la Figura 5.9), por lo que se concluye que las cantidades empleadas en el riego manual son muy bajas para los requerimientos de esta planta; segundo, es posible optimizar aun más la cantidad de agua utilizada por el sistema, dado que como se observó en la prueba anterior, existe un overshoot (exceso) que puede reducirse, abriendo las puertas a resultados aún más alentadores.

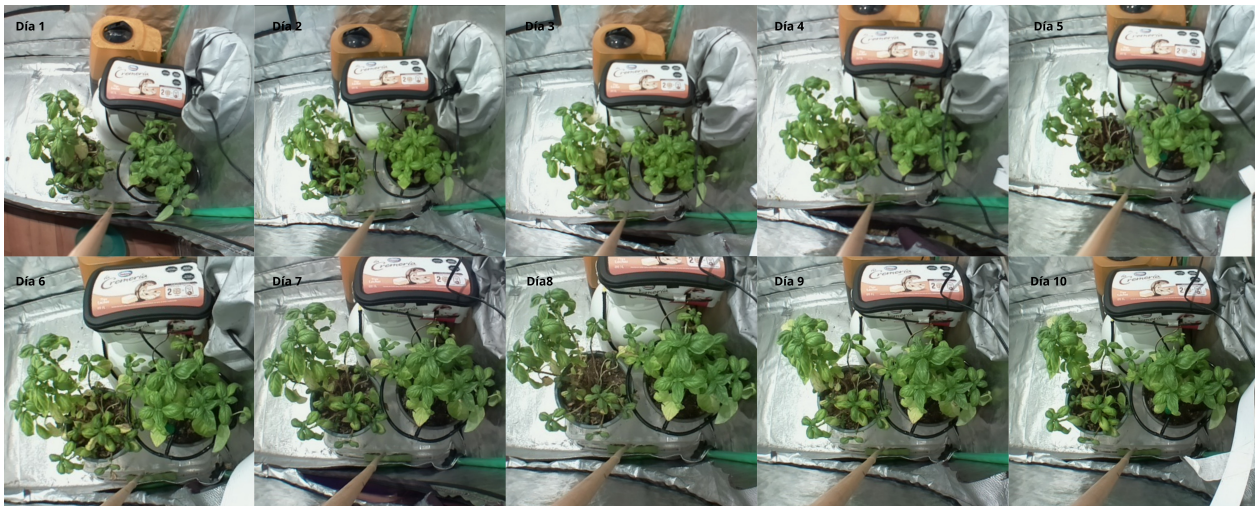


Figura 5.10: Timelapse de 10 días

Finalmente, la apariencia de la albahaca bajo riego manual (izquierda) terminó siendo bastante peor que la automatizada. A pesar de que en un periodo tan corto de tiempo no se pueda evidenciar esta alteración, si se pudo observar cómo bajo distintas situaciones cotidianas, el cuidado de esta planta era perjudicado. Por ejemplo, cada vez que se saltaban riegos o revisiones de su estado, o incluso cuando se hacían en un horario regular (como en esta prueba), ya podía apreciarse que tras un día sin riego la planta comenzaba a achicar y arrugar sus hojas, lo que claramente perjudica su crecimiento dado que genera un estrés innecesario (como se observa en la Figura 5.10).

5.2.5. Alertas y notificaciones

Esta última prueba consistió en analizar la correlación entre las últimas 50 notificaciones/alertas enviadas al dispositivo móvil, y la dinámica real del sistema. Se espera que cada cambio en los estados de los actuadores, como también las situaciones críticas definidas con anterioridad, sean entregadas al usuario en el mismo orden que ocurren en el sistema.

La Figura 5.11 muestra los últimos 5 mensajes recibidos en el dispositivo móvil (a través de Telegram), y a su derecha la misma cantidad de anotaciones en la interfaz gráfica, indicando cada una de estas acciones o situaciones críticas de las métricas, según el horario en que ocurrieron. Aunque éste es un extracto del total de mensajes analizados, permite ejemplificar el buen funcionamiento del sistema de alertas para el usuario. El resto de los mensajes analizados (45), se encuentran igualmente ordenados en su totalidad, por lo que se considera que el sistema de alertas y notificaciones funciona según lo esperado.

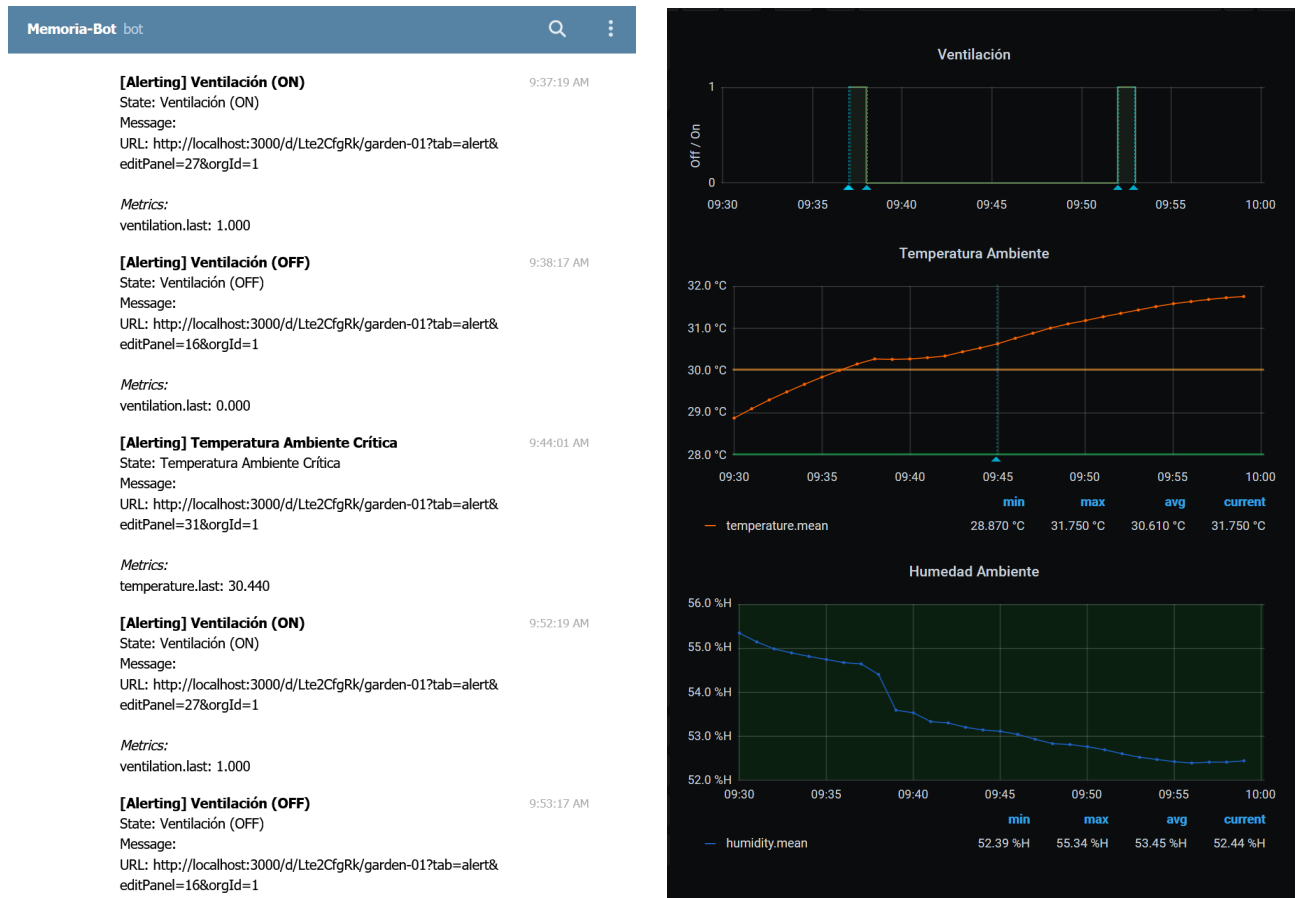


Figura 5.11: Extracto de las últimas 5 notificaciones al usuario

Capítulo 6

Conclusiones y trabajo a futuro

La concepción de este tipo de sistemas de asistencia al cultivo urbano, proviene de la experiencia común entre cultivadores que han evidenciado un malgasto de tiempo y recursos en procesos repetitivos e ineficientes durante la actividad de cuidado de sus plantas. Este problema inspira la creación de soluciones adaptables a diversos escenarios de cultivo, que consideren aspectos ambientales, económicos, de espacio y de la especie a cultivar. Si bien las posibilidades son bastante amplias, el alcance de este trabajo de memoria se limita a huertos urbanos y periurbanos que cuenten con disponibilidad eléctrica, de agua y red WiFi. Para ello se diseñó e implementó un sistema que resuelve las problemáticas presentadas, y que además permite al cultivador monitorear y recibir notificaciones en tiempo real sobre diferentes situaciones en su cultivo, con la finalidad de entregar asistencia de alto nivel a todo tipo de usuarios.

El desarrollo de esta memoria incorpora elementos de hardware y software, utilizando dispositivos electrónicos Open Source como son las placas Raspberry Pi y NodeMCU. Los programas escritos para integrar estos elementos con sus periféricos, como también aquellos que generan la interfaz del usuario, son diseñados a partir de tecnologías y librerías abiertas, que facilitan la implementación de este sistema de bajo costo, capaz de automatizar los procesos de riego y ventilación del cultivo. Utilizando estos dispositivos, se logra controlar cuatro variables esenciales en el desarrollo de las plantas: humedad y temperatura del sustrato, y del ambiente. Éstas serán reguladas en base a rangos de caracterización definidos por el usuario, personalizando así el funcionamiento automatizado del sistema según su propia experiencia y conocimientos.

El aspecto innovador de este sistema es la integración de dos conceptos y/o tecnologías modernas: Internet de las Cosas (IoT) y control difuso. El primero, sienta las bases para el diseño e implementación de los métodos de comunicación entre nuestros dispositivos y programas. El segundo, entrega una forma inteligente de transformar los valores recolectados por los sensores, en acciones directas y eficientes para su control. En específico, se utiliza el protocolo llamado Mosquitto (basado en MQTT) para generar las redes de comunicación que permitirán: 1) la transmisión de los valores recolectados y de las acciones de control, 2) la recolección (Paho MQTT) y almacenamiento de esta información en una base de datos en serie de tiempo (InfluxDB), y 3) la generación de las visualizaciones (Grafana) y alertas

(Telegram) antes descritas.

A primera vista, se confirma que las acciones realizadas por el sistema entregan resultados de control favorables y precisos. Para confirmar estas impresiones, se llevaron a cabo una serie de pruebas, como por ejemplo, la manipulación de las variables para realizar transiciones entre diversos estados, evaluaciones a corto y largo plazo, y comparaciones de los procesos automatizados con sus equivalentes de forma manual. Los resultados obtenidos se ajustan al comportamiento deseado, logrando tasas de error y desviación bastante bajas, además de reducciones considerables en el tiempo y recursos invertidos en el cultivo. Tomando en consideración las limitaciones impuestas y los posibles ajustes a métodos y parámetros del sistema indicados a lo largo de esta memoria, se concluye de forma exitosa la implementación del sistema de control presentado.

A lo largo de esta memoria se proponen una serie de modificaciones al controlador difuso, que buscan mejorar (o al menos comparar) los resultados ante las variaciones en sus componentes. Se pueden incluir a estas algunas funcionalidades importantes, como son la capacidad de modificar el estado de los actuadores directamente, limitar la acción de estos a horarios específicos del día, al estado de otras variables (por ejemplo, si se encuentran dentro de su rango letal) o a otro tipo de condiciones. También se puede incorporar el cambio de los parámetros del controlador y sus variables a la interfaz del usuario, y finalmente, se podría realizar una comparación con otros tipos de control avanzado.

Además de lo antes mencionado, es importante proyectar el perfeccionamiento del sistema en dos aspectos clave: la usabilidad y la escalabilidad. Un diseño local como el propuesto, trae como consecuencias la fragilidad ante fallas inesperadas en la red WiFi y red eléctrica. Adicionalmente, se deben considerar las carencias en la calidad y cantidad de sensores/actuadores disponibles, sumado a la falta de herramientas para asegurar su calibración y buen funcionamiento. Para abordar estas limitaciones se propone como solución: 1) la creación de redes locales de seguridad (utilizando la Raspberry Pi como Access Point y la NodeMCU con una memoria propia), 2) la evaluación y cotización de nuevos dispositivos para mejorar el diseño y robustez del sistema (sensor CO₂, nivel de agua, baterías, casing, componentes de seguridad, etc.), y 3) la implementación de estos servicios y sistemas en la nube, para así acceder de forma remota a la interfaz y delegar las responsabilidades de estabilidad y disponibilidad al servidor.

Para concluir, se plantea la integración de estos resultados al trabajo realizado por el Sr. Rodrigo Soria [27], quien desarrolló una aplicación móvil para cultivadores y comunidades de cultivo. Esto se traduce en la extensión de las funcionalidades desarrolladas en dicha memoria, estandarizando los procesos de gestión de ambientes de cultivo urbano, la creación y actualización de los elementos del ambiente de cultivo (plantas, ambientes, usuarios, etc.), entre otras cosas.

Bibliografía

- [1] César Mazquiarán Andrade. Control climático de un invernadero mediante lógica borrosa. Trabajo de fin de grado de ingeniería electrónica, industrial y automática. Universidad Politécnica de Madrid, España. Disponible en: http://oa.upm.es/47865/1/TFG_CESAR_MAZQUIARAN_ANDRADE.pdf. 2017.
- [2] Aspen Grow Box. <https://cannabistraininguniversity.com/blog/growing-marijuana/aspen-grow-box-an-adventure-in-home-hydroponics>, Última visita: 22 de Marzo de 2021.
- [3] Rachmad Atmoko, R Riantini, and M Hasin. Iot real time data acquisition using mqtt protocol. *Journal of Physics: Conference Series*, 853:012003, 05 2017.
- [4] José Azcue-Puma, Alfeu Sguarezi, and Ernesto Filho. Three types of fuzzy controllers applied in high-performance electric drives for three-phase induction motors. *In book: Fuzzy Controllers- Recent Advances in Theory and Applications*, 09 2012.
- [5] Troy Buechel. Air porosity: What is it and how important is it? *Disponible en: <https://www.pthorticulture.com/en/training-center/air-porosity-what-is-it-and-how-important-is-it/>*, 2020.
- [6] Hung-Yunn Chung, Bor-Chin Chen, and Jin-Jye Lin. A pi-type fuzzy controller with self-tuning scaling factors. *Fuzzy Sets and Systems*, 93(1):23–28, 1998.
- [7] Cloudponics. <https://cloudponics.com>, Última visita: 22 de Marzo de 2021.
- [8] Console Thinger.io. <https://console.thinger.io/>, Última visita: 22 de Marzo de 2021.
- [9] Christina Curell. Why is soil water holding capacity important? Disponible en: https://www.canr.msu.edu/news/why_is_soil_water_holding_capacity_important. Última visita: 22 de marzo de 2021.
- [10] Christopher David. Series of articles for mqtt. Disponible en: <https://diyiot.com/visualize-mqtt-data-with-influxdb-and-grafana/>. Última visita: 22 de marzo de 2021.
- [11] Eclipse Paho MQTT. <https://www.eclipse.org/paho/>, Última visita: 22 de Marzo de 2021.
- [12] Grafana. <https://grafana.com>, Última visita: 22 de Marzo de 2021.

- [13] Samuel Greengard. *The Internet of Things*. The MIT Press, 2015.
- [14] Groupal Technology. <https://www.growpaltech.com/>, Última visita: 22 de Marzo de 2021.
- [15] Ronald L. Hanson. Evapotranspiration and droughts. *In: U.S. Geological Survey Water-Supply. Paper 2375, p. 99-104*, 1991.
- [16] InfluxDB. <https://www.influxdata.com/>, Última visita: 22 de Marzo de 2021.
- [17] Robert Kandrsmith. Wide range of hygrometers: Dht22, am2302, am2320, am2321, sht71, htu21d, si7021, bme280. *Disponible en: https://www.kandrsmith.org/RJS/Misc/Hygrometers/calib_many.html*, 2018.
- [18] M.B. Kirkham. *Principles of Soil and Plant Water Relations*. Elsevier. 2005.
- [19] Roger A. Light. Mosquitto: server and client implementation of the mqtt protocol. *Journal of Open Source Software*, 2(13):265, 2017.
- [20] E.H. Mamdani and S. Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies*, 7(1):1–13, 1975.
- [21] Samuel Moya. Conceptos básicos: Sistemas de control. *Disponible en: https://www.isamex.org/intechmx/index.php/2018/12/24/conceptos-basicos-sistemas-de-control/*, 2018.
- [22] David Christian Rose and Jason Chilvers. Agriculture 4.0: Broadening responsible innovation in an era of smart farming. *Frontiers in Sustainable Food Systems*, 2:87, 2018.
- [23] Masaru Sakamoto and Takahiro Suzuki. Effect of root-zone temperature on growth and quality of hydroponically grown red leaf lettuce. *American Journal of Plant Sciences*, 6, 2350-2360, 2015.
- [24] Scikit-Fuzzy. Defuzzification. *Disponible en: https://pythonhosted.org/scikit-fuzzy/auto_examples/plot_defuzzify.html*, 2020.
- [25] Scikit Fuzzy en Python. <https://pythonhosted.org/scikit-fuzzy>, Última visita: 22 de Marzo de 2021.
- [26] Vasudha Sharma and Jerry Wright. Irrigation management strategies. *Disponible en: https://extension.umn.edu/irrigation/irrigation-management-strategies#early-season-strategies-1702910*, 2019.
- [27] Rodrigo Soria. Aplicación de gestión semiautomática de cultivos urbanos. *Memoria de Ingeniería Civil en Computación, FCFM, Universidad de Chile*, 2021.
- [28] OASIS Standard. Mqtt:the standard for iot messaging. *MQTT Specification version 5.0*, 2019.
- [29] Doris Sáez. *Apunte de Control Avanzado de Sistema*. Departamento de Ingeniería Eléc-

trica, FCFM, Universidad de Chile. 2019.

- [30] Telegram. Bots: An introduction for developers. *Disponible en: <https://core.telegram.org/bots>*, 2021.
- [31] Thinger.io. <https://www.thinger.io/>, Última visita: 22 de Marzo de 2021.
- [32] Javier Vivanco. El modelo c4 de documentación para la arquitectura de software. *Disponible en: <https://medium.com/@javiervivanco/el-modelo-c4-de-documentacion-para-la-arquitectura-de-software-424704528390>*, 2019.
- [33] Francisco Javier Vásquez. Justicia en los ríos: La lucha del movimiento por el acceso al agua, la tierra y la protección del medio ambiente en la provincia de petorca. Universidad de Playa Ancha, Chile. *Disponible en: http://modatima.cl/wp-content/uploads/2018/04/FcoVazquez2012_Justicia_en_los_rios_La_lucha_del_MODATIMA_Provincia_de_Petorca.pdf*, 2012.
- [34] Mark Weiser. The computer for the 21st century. *Scientific American*, 265(3):66–75, January 1991.
- [35] Wise and Growth. <http://www.facebook.com/wisegrowth.io>, Última visita: 22 de Marzo de 2021.
- [36] Vasudha Sharmaand Jerry Wright. Basics of irrigation scheduling. *Disponible en: <https://extension.umn.edu/irrigation/basics-irrigation-scheduling>*, 2019.
- [37] Feng Xia, Laurence T Yang, Lizhe Wang, and Alexey Vinel. Internet of things. *International journal of communication systems*, 25(9):1101, 2012.