



# Tecnológico de Monterrey

Implementación del Internet de las cosas

Sebastián Acosta Marín - A01278278

Yessica Lora Vazquez - A01273574

Monica Sarahí Flores Lara - A07045944

Gabriel Esperilla León - A01277955

Fecha de entrega: 28 de noviembre de 2024

## Objetivos

En la agricultura convencional, el control de las variables catalogadas como críticas, como lo son la temperatura, humedad y la radiación solar, puede llegar a ser un problema, y puede convertirse en un proceso impreciso o mal controlado, lo cual puede llevar a un uso indebido, inadecuado e impreciso de recursos esenciales para el cuidado de la maceta como lo puede ser el agua y la energía; En el cultivo de la planta de menta, un monitoreo pobre e ineficiente puede causar mucho estrés en las plantas, afectando de esta manera su crecimiento y su producción, por lo que la problemática que busca atender este proyecto, es el pobre monitoreo de las plantas de cultivo.

Este proyecto tiene como objetivo principal diseñar y desarrollar un sistema automatizado que sea capaz de monitorizar las condiciones ambientales que pueden llegar a afectar a una planta en una maceta tradicional. Este sistema automatizado utiliza sensores que medirán las variables importantes las cuales son temperatura, humedad y radiación solar; estas mediciones se realizan con la finalidad de optimizar el riego, la exposición a la luz solar y cuidar la temperatura.

De la mano de todo esto, otro de nuestros objetivos es mejorar y optimizar el crecimiento y bienestar de la planta.

Originalmente dentro de nuestros objetivos se tenía la idea de realizar una comparación exhaustiva en la que se pudiera visualizar de manera clara y contundente las ventajas de usar una maceta asistida por el IoT en comparación de una maceta tradicional, pero debido a que nos quedamos sin tiempo, no nos fue posible realizar el registro de ambas macetas.

## Planta seleccionada (Menta)

La planta que como equipo seleccionamos fue la de menta debido a las siguientes razones:

- Es fácil de cultivar y mantener
  - Cuenta con resiliencia
  - Cuenta con adaptabilidad
- Es compatible con nuestro proyecto de IoT
  - Cuenta con variables bien establecidas
    - Temperatura:
      - **Rango óptimo:** 15°C a 25°C.
      - **Sensor recomendado:** DS18B20 (precisos y económicos).
    - Humedad
      - **Rango óptimo:** 50% - 70%
      - **Sensor recomendado:** Sensor capacitivo de humedad del suelo V2.0
    - Radiación
      - **Rango óptimo:** 10,000 - 25,000 lux
      - **Sensor recomendado:** BH1750
  - Automatización sencilla
  - Datos constantes
  - Fácil de monitorear
  - respuestas de la planta tangibles
- Tiene múltiples propósitos
  - Usos múltiples (Culinarias, medicinales, aromáticas)

## Marco teórico

Para llevar a cabo nuestro proyecto, hicimos las mediciones de las variables que consideramos críticas, las cuales son:

- Radiación solar
- Temperatura de la tierra
- Humedad

Para realizar estas mediciones lo que utilizamos fue:

### **Sensor capacitivo de humedad del suelo V2.0**

El funcionamiento del sensor capacitivo de humedad tiene su base en la medición de la capacitancia que se da entre dos electrodos que se encuentran enterrados debajo de la tierra, la capacitancia que se da entre esos dos electrodos dependerá completamente de la humedad que hay en la tierra; esto se debe a que la tierra seca cuenta con una permitividad baja, lo que significa que la tierra no contribuye bastante a la capacitancia. Sin embargo y por lo contrario el agua tiene una permitividad alta, por lo que cuando la tierra está húmeda, la presencia de agua entre los electrodos actúa como un dieléctrico que aumenta de manera significativa la capacitancia del sensor, por lo que a mayor capacitancia, existe una mayor humedad en la tierra. El electrodo va conectado a una tarjeta de acondicionamiento que entrega una salida analógica, es decir, que transforma la señal del electrodo en una señal analógica limpia y estable la cual es sencilla de interpretar para otros dispositivos, como lo son los microcontroladores; Esta salida analógica entrega un voltaje que varía de los 0V hasta los 5V dependiendo de la humedad de la tierra (0V para tierra seca y 5V para tierra muy húmeda)

Este sensor cuenta con especificaciones técnicas las cuales son:

- El voltaje de alimentación con la cual trabaja el sensor de humedad debe oscilar entre los 3.3V y los 5V (compatible con arduino y con la ESP32) es importante que se tenga en cuenta el voltaje todo el tiempo que se esté

realizando pruebas con el sensor debido a que si no se respeta el voltaje, el funcionamiento podría volverse inestable, es decir, errores de mediciones

- La corriente de operación es de 5 mA, es decir, que cada vez que el sensor se encuentre funcionando esté consume un total de 5 mA de corriente eléctrica, esto es bueno debido a que esta corriente es relativamente baja, y puede ser fácilmente proporcionada por un puerto USB asegurando así que el sensor funciones de manera segura y confiable.
- El sensor cuenta con una vida útil mínima de 3 años; esto nos sirve de manera excelente ya que nuestro proyecto a realizar no es a tan largo plazo, sino que consta simplemente de 10 semanas.
- El sensor cuenta con un conector PH2.0-3P, el cual utiliza para conectarse a otros componentes, como lo pueden ser una tarjeta de acondicionamiento o un microcontrolador.

### **Sensor DS18B20 (temperatura)**

El sensor DS18B20 es un sensor digital de temperatura que utiliza el protocolo 1-Wire para comunicarse, es decir que solo necesita de solo pin de datos para poder comunicarse. El sensor es útil para medir temperaturas que van desde los -55°C hasta los 125°C es cual es un intervalo bastante amplio ya que en la naturaleza de nuestro proyecto no se trabajará con temperaturas tan extremas. El sensor es resistente al agua y de igual manera es compatible con arduino y con la ESP32, y además internamente se encuentra sensado y acondicionado para convertir una señal analógica a digital e interfaz digital. - Para que el sensor DS18B20 pueda funcionar con arduino es necesario que se instalen dos librerías: - OneWire: En esta librería se encuentra implementado todo lo que tiene que ver con el protocolo con el que funciona el sensor - DallasTemperature: En esta librería están implementadas todas las funciones necesarias para que el sensor pueda realizar las lecturas de las temperaturas de manera correcta

Las especificaciones técnicas de este sensor son:

- El voltaje de alimentación con la cual trabaja el sensor de temperatura debe oscilar entre los 3.0V y los 5.5V (compatible con arduino y con la ESP32) es

importante que se tenga en cuenta el voltaje todo el tiempo que se esté realizando pruebas con el sensor debido a que si no se respeta el voltaje, el funcionamiento podría volverse inestable, es decir, errores de mediciones

- El sensor cuenta con un rango de medición que va desde los -55°C hasta los 125°C lo cual es un rango de medición bastante alto - En el intervalo de medición que va desde los -10°C hasta los 85°C la precisión de la medición es de  $\pm 5^{\circ}\text{C}$
- Cuenta con una resolución ADC seleccionable que puede ir desde los 9 a los 12 bits, lo cual nos permite ajustar la precisión y el rango dinámico al momento de realizar las lecturas analógicas.
- Cuenta con tres cables: -
  - Rojo: +VCC
  - Blanco: DATA 1-Wire
  - Negro: Ground (positive supply voltage ) -
- El sensor tiene una identificación única de 64 bits lo cual le otorga una referencia única e irrepetible, por lo que gracias a esto, es posible conectar más de un sensor en un mismo bus, es decir, tener conectados sensores múltiples al mismo canal de comunicación, compartiendo así las mismas líneas de datos en lugar de usar una conexión independiente por sensor; esto se debe a que el sensor funciona de la mano con el protocolo de comunicación 1-Wire
- El sensor cuenta con una cubierta de acero inoxidable, con la finalidad de evitar que la onda se oxide
- La sonda es a prueba de agua, en caso de que se necesiten realizar pruebas en el agua
- La longitud del cable es de 1 metro

### **BH1750 (luz solar)**

El sensor BH1750 es un sensor de iluminación digital que sirve para medir el flujo luminoso denominado iluminancia. Cuenta con un conversor interno de 16 bits, y es gracias a esto que es capaz de entregar una salida digital en formato I2C (IICInter-Integrated Circuit) el cual es un protocolo que utiliza dos líneas para poder

controlar otros dispositivos, y permite que varios dispositivos se comuniquen en un mismo bus; y además al utilizar este protocolo de comunicación, es compatible tanto con arduino como con la ESP32. Los dos cables son los SDA(línea de datos) y el SCL( línea de reloj) los cuales facilitan la comunicación mediante la transmisión de los datos de manera sincronizada y con pulsos de reloj. El sensor entrega su mediciones directamente en unidades de Lux la cual es la unidad del sistema internacional para medir la cantidad de luz que se proyecta en una superficie por lo que no es necesario realizar un factor de conversión

Las especificaciones técnicas para este sensor son:

- El sensor cuenta con un voltaje de operación que puede oscilar entre los 3V y los 5V de corriente directa
- Cuenta con un módulo GY-302 que sirve para medir la intensidad de la luz ambiente de forma digital y además lo hace de manera precisa.
- El sensor también cuenta con una interfaz digital que al utilizar el protocolo de IIC es capaz de seleccionar entre dos direcciones de comunicación posibles
- El sensor realiza mediciones en un rango que va desde 1 a 65,535 lux, lo cual es un nivel de luz extremadamente alto, haciendo así que el sensor sea bastante útil
- Cuenta con un rechazo de ruido a 50/60 Hz lo cual quiere decir que ignora la luz que proviene de fuentes de luz parpadeantes o de baja calidad

## **Calibración de los sensores**

### **Sensor capacitivo de humedad del suelo V2.0**

Para calibrar el sensor de humedad, realizamos pruebas en dos condiciones distintas: en aire y sumergido en agua. Estos valores extremos se usaron como referencia, asignando un valor de 0% de humedad cuando el sensor estaba en aire (indicación de suelo seco) y 100% cuando estaba sumergido (indicación de suelo completamente húmedo). Esta calibración permite obtener una medición confiable de la humedad del suelo.

### **Sensor DS18B20 (temperatura)**

Se utilizó la temperatura ambiente como referencia. Realizamos mediciones en condiciones de ambiente estable, observando que las lecturas variaban en un rango muy reducido, apenas entre medio grado y un grado. Esta estabilidad y precisión confirmaron que el sensor estaba funcionando correctamente y que no requería ajustes adicionales.

### **BH1750 (luz solar)**

Se empleó la linterna de un teléfono móvil como fuente de luz controlada. Buscamos en Internet el valor aproximado de la intensidad de luz de la linterna del teléfono y ajustamos el sensor hasta que mostrará un valor cercano al conocido. Con esto logramos una calibración precisa para el sensor, permitiendo que midiera la intensidad de luz de forma confiable y en la escala esperada.

## **Actuadores**

El proyecto se centrará en el control manual asistido por los actuadores, los cuales se gestionan automáticamente mediante una válvula de riego, un ventilador y un servomotor que abrirá una compuerta. Este enfoque busca garantizar que el usuario mantenga control directo sobre las decisiones importantes, con el apoyo de un



sistema de monitoreo que facilite la detección de cambios o necesidades en tiempo real.

Para suministrar energía a toda la protoboard se emplearon unas pilas y un portapilas

#### **Porta pilas de 2 pilas 18650:**

- **Configuración de pilas:** 2 pilas 18650 en **serie**, lo que proporciona un voltaje de salida total de **7.4V** (3.7V por celda, en serie).
- **Conector:** Viene con un conector **DC Jack de 5.5x2.1mm**, que es comúnmente utilizado en proyectos con **Arduino** y otros dispositivos electrónicos, con el positivo en el interior y el negativo en el exterior.
- **Material:** Fabricado en **plástico resistente**, diseñado para mantener las pilas en su lugar de manera segura.
- **Tamaño:** Compacto y adecuado para proyectos pequeños o medianos, como **seguidores de línea** o **carros evasores de obstáculos**.
- **Montaje:** Tiene **orificios para tornillos** en la base, permitiendo fijarlo fácilmente a una superficie.
- **Longitud del cable:** Aproximadamente **15 cm**, que permite una instalación cómoda en proyectos donde se necesita conexión rápida.
- **Compatibilidad:** Ideal para proyectos que requieren una fuente de energía portátil y ligera, con un voltaje de 7.4V.

### **1. Control Automático de Humedad (Riego Automatizado)**

**Rango óptimo: 50% - 70%**

El riego se gestionará mediante un sensor capacitivo de humedad del suelo y un actuador tipo electroválvula:

- **Funcionamiento:** Cuando el sensor detecta niveles bajos de humedad, la electroválvula se abre para permitir el flujo de agua. Una vez alcanzado el nivel de humedad óptimo, la válvula se cierra automáticamente.

Componentes involucrados:

- Sensor capacitivo conectado al microcontrolador ESP32 para leer datos y activar la válvula.

### **Mini Bomba de Agua 3v-6v Sumergible**

La mini bomba de agua sumergible de 3V-6V es ideal para proyectos pequeños como fuentes, sistemas de riego o refrigeración. Aquí tienes sus especificaciones técnicas principales:

- Voltaje de operación: 3V a 6V.
- Consumo de corriente: Aproximadamente 300 mA.
- Caudal máximo: 80-120 litros por hora (alrededor de 2 litros por minuto).
- Elevación máxima: Entre 40 y 110 cm.
- Tamaño de salida: 7.5 mm.
- Tamaño de entrada: 5 mm.
- Diámetro del dispositivo: 24 mm, con una altura aproximada de 30 mm y una longitud de 45 mm.
- Material: Plástico ABS, resistente al agua y a la erosión.
- Vida útil: 500 horas de trabajo continuo.
- Modo de funcionamiento: Motor sin escobillas, con conducción magnética.

## **2. Actuador para el Control de Temperatura (Ventilación automática)**

**Rango óptimo: 15°C a 25°C.**

El control de la temperatura se gestionará de manera asistida por un sensor DS18B20 que activará un ventilador en caso de que la temperatura no sea la óptima  
Componentes:

- Sensor DS18B20 para monitoreo continuo.

Un ventilador de 5V para Raspberry Pi está diseñado para proporcionar refrigeración eficiente en proyectos con microcomputadoras. Aquí tienes sus especificaciones técnicas típicas:

- **Voltaje de alimentación:** 5V DC.
- **Corriente nominal:** 130 mA.
- **Dimensiones:** 30 x 30 x 10 mm (algunas variantes pueden tener tamaños diferentes).
- **Peso:** Aproximadamente 10 g.
- **Longitud del cable:** 28 cm.
- **Motor:** Sin escobillas, para mayor durabilidad y menor ruido

### **3. Actuador para el Control de Luz Solar (Compuerta automática)**

**Rango óptimo: 10,000 - 25,000 lux**

El sistema de luz monitorea la intensidad solar mediante el sensor BH1750. Cuando se detecte un nivel de luz solar excesiva, el sistema activará un servomotor de manera automática, el cual cerrará una compuerta para evitar el paso de la radiación solar, en caso contrario, la compuerta se abrirá para permitir el paso de la luz.

Componentes:

- Sensor BH1750 para medición precisa de luz en lux.

#### **Servomotor SG90 RC 9g**

El servomotor SG90 RC 9g es un pequeño motor de precisión ampliamente utilizado en proyectos de robótica, drones, y sistemas controlados por microcontroladores como Arduino. Aquí están sus especificaciones principales:

- **Dimensiones:** 22.8 mm × 12.3 mm × 22.5 mm.
- **Peso:** 9 g (10.6 g con cable y conector).
- **Voltaje de operación:** 4.8V a 6V (se recomienda 5V para mejor rendimiento).
- **Ángulo de rotación:** 0° a 180°.
- **Torque:** Hasta 1.2 kg/cm a 4.8V y 1.6 kg/cm a 6V.
- **Velocidad:** 0.12 segundos para un giro de 60° a 4.8V.
- **Engranajes:** Nylon.
- **Conexión:** Cable de 3 pines: rojo (alimentación), marrón (tierra) y naranja (señal PWM).
- **Temperatura de operación:** -30°C a 60°C

## Metodología

Hemos decidido utilizar la metodología Scrumban en nuestro proyecto porque combina la flexibilidad de Kanban con la estructura y planificación de Scrum, adaptándose a nuestras necesidades que consideramos que con el tiempo pueden ir cambiando, esto ya que nuestro proyecto cuenta con múltiples fases y un alcance que puede requerir ajustes sobre la marcha, Scrumban nos permite organizar las tareas de manera eficiente, visualizar el flujo de trabajo y priorizar actividades críticas sin la rigidez de los sprints fijos de Scrum.

La implementación de la metodología Scrumban nos ayudó significativamente a llevar a cabo nuestro proyecto, ya que este enfoque fue clave para gestionar eficientemente las múltiples fases y el alcance variable del proyecto.

Esta metodología fue clave para:

- Organización de tareas
- adaptarse a cambios de trabajos
- Planificación estratégica
- Priorización efectiva
- Mejor comunicación

Es por eso que gracias a esta metodología logramos concluir nuestro proyecto dentro del tiempo estimado, con margen para iterar sobre partes críticas y asegurar una calidad óptima en cada fase. Este enfoque flexible pero estructurado fue esencial para manejar la complejidad y la evolución natural de las necesidades del proyecto.

Para llevar a cabo la metodología Scrumban en nuestro proyecto, decidimos utilizar Microsoft Planner como herramienta principal. Esta plataforma nos permitió gestionar nuestras tareas de manera eficiente, visualizando el flujo de trabajo y adaptándonos a las necesidades cambiantes del proyecto. Lo utilizamos debido a que nos sirve para:

- Visualización del flujo del trabajo
- Flexibilidad

- Priorización de tareas
- Asignación de tareas
- Colaboración en equipo
- Revisión y ajuste continuo.

The screenshot shows a Microsoft Planner board with the following structure:

- Por Hacer (To Do):** Contains a button '+ Agregar tarea'.
- En Proceso (In Progress):** Contains one task: 'Evaluación oral y escrita 1 FINAL'. Below it, a 'Vencimiento' (Due Date) section shows icons for 'YV', 'GL', 'M', and '+1'. At the bottom, it indicates 'Tareas completadas: 1'.
- Hecho (Done):** Contains 9 completed tasks:
  - Investigación y Pruebas de Sensores (Durante estas semanas se realizaron pruebas iniciales con los sensores seleccionados para garantizar su correcto funcionamiento)
  - Completada por Yessica Lora Váz...
  - Desarrollo del Código e Integración de Sensores (Se trabajó en la programación del sistema para que los sensores recojan y almacenen datos de manera automática)
  - Completada por Yessica Lora Váz...
  - Creación de Base de Datos y Página WEB
  - Completada por Yessica Lora Váz...
  - Montaje y Pruebas Finales (Ensamblaje completo del sistema en una maceta real para monitorear una planta. Se realizarán ajustes y pruebas finales para asegurar que el sistema esté listo para la presentación)
  - Completada por Yessica Lora Váz...
  - Ajustes del código y pruebas finales con todos los elementos integrados (Prueba de la interacción de todos los elementos para asegurar su correcto funcionamiento)
- Reuniones Semanales (Weekly Meetings):** Contains 8 completed weekly meetings, all marked as 'Completada por Yessica Lora Váz...'.

En la imagen, la única actividad en proceso es la evaluación oral y escrita, mientras que todas las demás tareas, incluidas las reuniones, están marcadas como completadas. Esto refleja el estado actual del proyecto, donde hemos logrado avanzar en muchas áreas, pero aún estamos en la etapa final de evaluación, que es un paso crucial para el cierre del ciclo.

## **ETAPA 1 - Semana 2**

En esta etapa, llevamos a cabo la definición completa del proyecto y sus alcances. Para ello, identificamos y detallamos los siguientes aspectos:

Nombre del proyecto

Área de investigación

Marco teórico

Descripción del problema a resolver

Lista de variables a medir y controlar

Esta etapa del reto la llevamos a cabo en un aproximado de 4 horas en las cuales, como equipo fuimos capaces de completar la primera etapa y con esto, la planificación de nuestro proyecto

## **ETAPA 2 - Semana 3**

En esta etapa, nos enfocamos en la planificación del proyecto, tomando como base la definición previa del mismo. Para llevar a cabo esta etapa del proyecto definimos los siguientes aspectos:

Organización del proyecto

Alcances del proyecto

Limitaciones de tiempo

Limitaciones de costo

Limitaciones de conocimiento

Esta etapa del reto la llevamos a cabo en un aproximado de 4 horas en las cuales, como equipo logramos crear una visión clara y realista de cómo proceder en las próximas fases del proyecto

## **ETAPA 3 - Semana 4**

En esta etapa, nos dedicamos a la investigación documental del proyecto. Después de haber definido y planificado el proyecto, destinamos 4 horas para investigar las posibles soluciones y herramientas necesarias para su implementación. Los aspectos que investigamos ya que considerábamos que eran clave son los siguientes:

Investigación de sensores

Investigación de actuadores Opciones para bases de datos

Opciones para páginas web/dashboard

## **ETAPA 4 - Semana 5**

La etapa 4 como equipo fuimos capaces de terminar toda la investigación y planificación realizada previamente de manera precisa y definir con precisión las variables físicas a medir, los sensores que se utilizarán, los actuadores y cómo estos interactúan en el ambiente de nuestro proyecto. Los puntos clave que abordamos incluyen:

Variables a medir

Tiempo de implementación

Funcionalidad del proyecto

Diseño del sitio web y base de datos

Plan de avance para las próximas 5 semanas

## **ETAPA 5 - Semana 6 a 9**

En esta etapa, realizamos diversas actividades clave relacionadas con la integración de sensores, actuadores y la conexión con Azure IoT. A continuación, se detallan los principales logros y actividades:

### **1. Calibración y Configuración de Sensores:**

- Conseguimos el código necesario para los sensores y procedimos a calibrarlos.
- Adaptamos los códigos a las variables específicas de nuestro proyecto.

### **2. Investigación sobre Azure IoT:**

- Nos enfocamos en cómo conectar los sensores a través de la ESP32 con Azure IoT.
- Estudiamos los códigos necesarios y las configuraciones para enviar los datos leídos hacia la base de datos en Azure.

### **3. Conexión y Pruebas:**

- Aunque el código para enviar datos a la base de datos en Azure era extenso y logramos vincular nuestros sensores con el IoT Dashboard de Azure, permitiendo la visualización de datos.

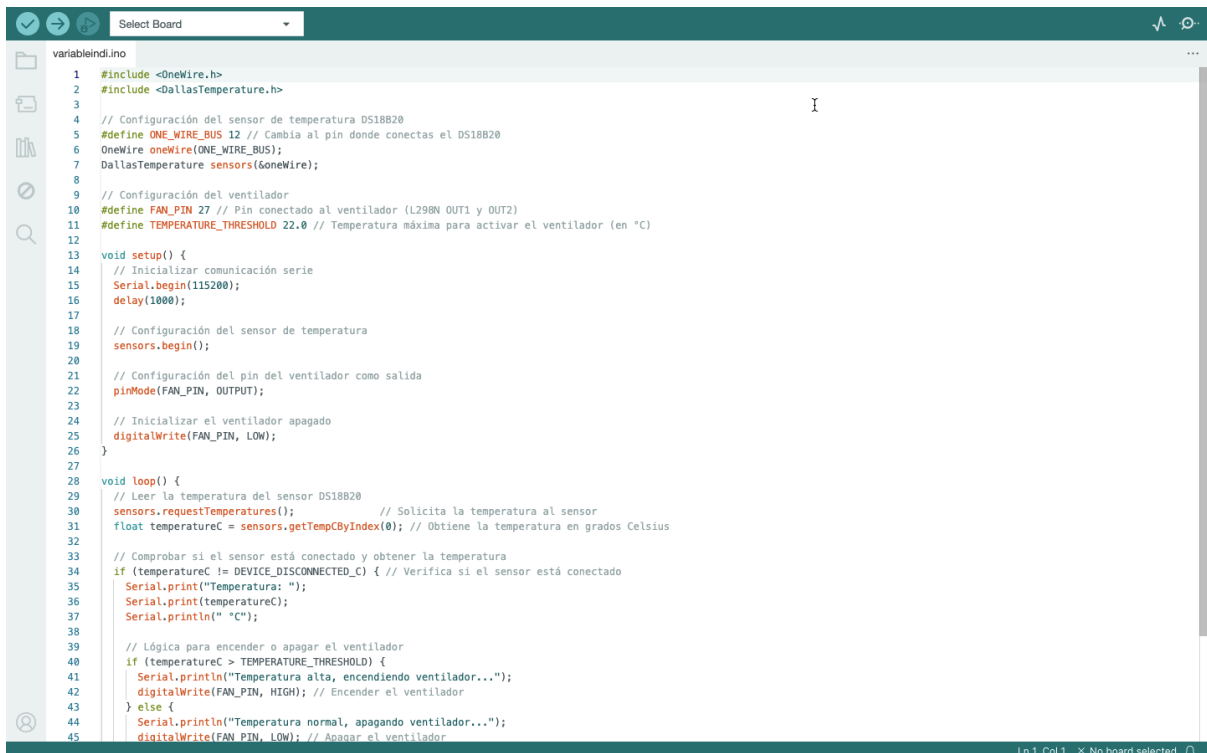


#### 4. Selección de Actuadores:

- Investigamos cuáles serían los actuadores adecuados para nuestro proyecto.
- Identificamos los componentes necesarios para comprar y comenzamos a planificar cómo integrarlos en el sistema.

#### 5. Implementación del Código para Actuadores:

- Investigamos y adaptamos el código para controlar actuadores como bombas de agua y ventiladores, con base en los datos obtenidos de los sensores.
- Aseguramos que los actuadores respondieron automáticamente a las variables según los umbrales definidos.
- Diseñamos la lógica necesaria para que los actuadores trabajen en conjunto con los sensores y se integren con el sistema de IoT.



```
1 #include <OneWire.h>
2 #include <DallasTemperature.h>
3
4 // Configuración del sensor de temperatura DS18B20
5 #define ONE_WIRE_BUS 12 // Cambia al pin donde conectas el DS18B20
6 OneWire oneWire(ONE_WIRE_BUS);
7 DallasTemperature sensors(&oneWire);
8
9 // Configuración del ventilador
10 #define FAN_PIN 27 // Pin conectado al ventilador (L298N OUT1 y OUT2)
11 #define TEMPERATURE_THRESHOLD 22.0 // Temperatura máxima para activar el ventilador (en °C)
12
13 void setup() {
14   // Inicializar comunicación serie
15   Serial.begin(115200);
16   delay(1000);
17
18   // Configuración del sensor de temperatura
19   sensors.begin();
20
21   // Configuración del pin del ventilador como salida
22   pinMode(FAN_PIN, OUTPUT);
23
24   // Inicializar el ventilador apagado
25   digitalWrite(FAN_PIN, LOW);
26 }
27
28 void loop() {
29   // Leer la temperatura del sensor DS18B20
30   sensors.requestTemperatures(); // Solicita la temperatura al sensor
31   float temperatureC = sensors.getTempCByIndex(0); // Obtiene la temperatura en grados Celsius
32
33   // Comprobar si el sensor está conectado y obtener la temperatura
34   if (temperatureC != DEVICE_DISCONNECTED_C) { // Verifica si el sensor está conectado
35     Serial.print("Temperatura: ");
36     Serial.print(temperatureC);
37     Serial.println(" °C");
38
39     // Lógica para encender o apagar el ventilador
40     if (temperatureC > TEMPERATURE_THRESHOLD) {
41       Serial.println("Temperatura alta, encendiendo ventilador...");
42       digitalWrite(FAN_PIN, HIGH); // Encender el ventilador
43     } else {
44       Serial.println("Temperatura normal, apagando ventilador...");
45       digitalWrite(FAN_PIN, LOW); // Apagar el ventilador
46     }
47   }
48 }
```

```

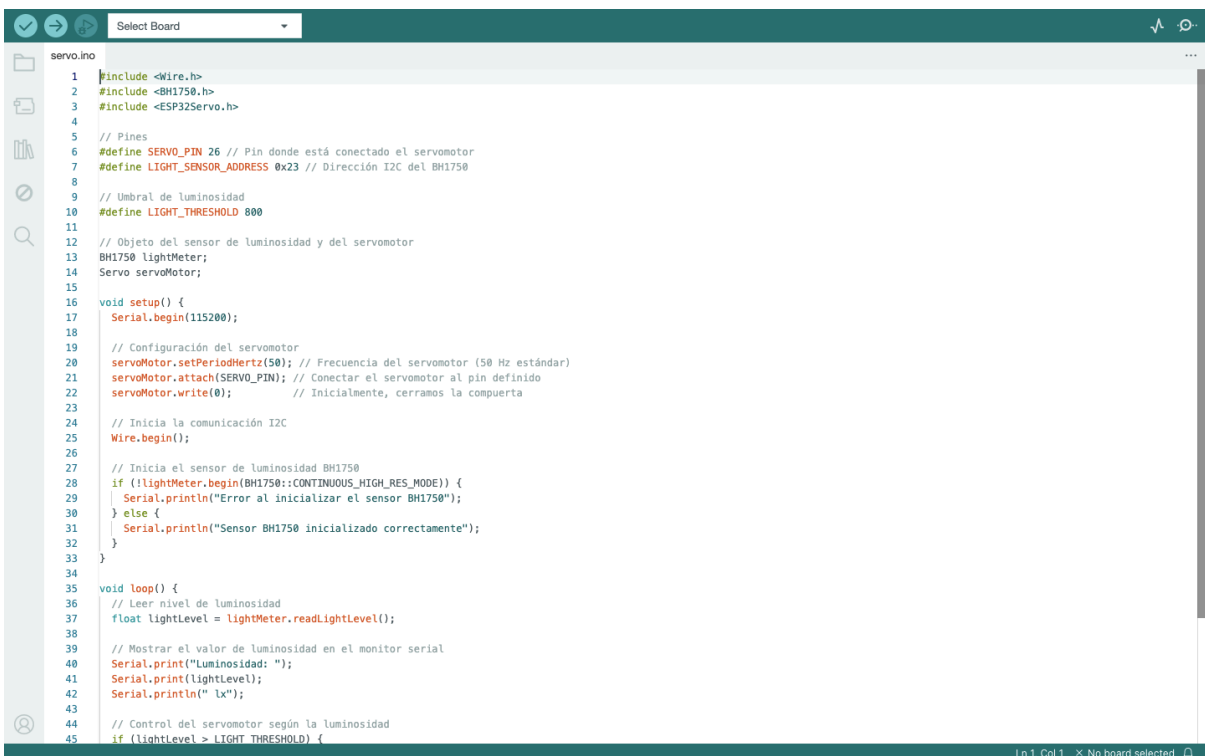
// Lógica para encender o apagar el ventilador
if (temperatureC > TEMPERATURE_THRESHOLD) {
    Serial.println("Temperatura alta, encendiendo ventilador...");
    digitalWrite(FAN_PIN, HIGH); // Encender el ventilador
} else {
    Serial.println("Temperatura normal, apagando ventilador...");
    digitalWrite(FAN_PIN, LOW); // Apagar el ventilador
}

```

```

Temperatura normal, apagando ventilador...
Temperatura: 21.06 °C
Temperatura normal, apagando ventilador...
Temperatura: 21.81 °C
Temperatura normal, apagando ventilador...
Temperatura: 22.56 °C
Temperatura alta, encendiendo ventilador...
Temperatura: 23.00 °C
Temperatura alta, encendiendo ventilador...
Temperatura: 23.44 °C
Temperatura alta, encendiendo ventilador...
Temperatura: 23.69 °C
Temperatura alta, encendiendo ventilador...

```



```

servo.ino
1 #include <Wire.h>
2 #include <BH1750.h>
3 #include <ESP32Servo.h>
4
5 // Pines
6 #define SERVO_PIN 26 // Pin donde está conectado el servomotor
7 #define LIGHT_SENSOR_ADDRESS 0x23 // Dirección I2C del BH1750
8
9 // Umbral de luminosidad
10 #define LIGHT_THRESHOLD 800
11
12 // Objeto del sensor de luminosidad y del servomotor
13 BH1750 lightMeter;
14 Servo servoMotor;
15
16 void setup() {
17     Serial.begin(115200);
18
19     // Configuración del servomotor
20     servoMotor.setPeriodHertz(50); // Frecuencia del servomotor (50 Hz estándar)
21     servoMotor.attach(SERVO_PIN); // Conectar el servomotor al pin definido
22     servoMotor.write(0); // Inicialmente, cerramos la compuerta
23
24     // Inicia la comunicación I2C
25     Wire.begin();
26
27     // Inicia el sensor de luminosidad BH1750
28     if (!lightMeter.begin(BH1750::CONTINUOUS_HIGH_RES_MODE)) {
29         Serial.println("Error al inicializar el sensor BH1750");
30     } else {
31         Serial.println("Sensor BH1750 inicializado correctamente");
32     }
33 }
34
35 void loop() {
36     // Leer nivel de luminosidad
37     float lightLevel = lightMeter.readLightLevel();
38
39     // Mostrar el valor de luminosidad en el monitor serial
40     Serial.print("Luminosidad: ");
41     Serial.print(lightLevel);
42     Serial.println(" lx");
43
44     // Control del servomotor según la luminosidad
45     if (lightLevel > LIGHT_THRESHOLD) {

```

```
Select Board
actuaadores_jot.ino
1 #include <Wire.h>
2 #include <BH1750.h>
3 #include <ESP32Servo.h>
4 #include <OneWire.h>
5 #include <DallasTemperature.h>
6
7 // ** Configuración del servomotor **
8 #define SERVO_PIN 26 // Pin donde está conectado el servomotor
9 #define LIGHT_SENSOR_ADDRESS 0x23 // Dirección I2C del BH1750
10 #define LIGHT_THRESHOLD 2000 // Umbral de luminosidad
11
12 BH1750 lightMeter;
13 Servo servoMotor;
14
15 // ** Configuración del ventilador **
16 #define FAN_PIN 27 // Pin conectado al ventilador (L298N OUT1 y OUT2)
17 #define TEMPERATURE_THRESHOLD 25.0 // Temperatura máxima para activar el ventilador (en °C)
18
19 OneWire oneWire(2); // Pin donde conectas el DS18B20
20 DallasTemperature sensors(&oneWire);
21
22 // ** Configuración de la bomba de agua **
23 #define IN1_PIN 25 // Pin conectado a IN1 del L298N
24 #define IN2_PIN 33 // Pin conectado a IN2 del L298N
25 #define SOIL_MOISTURE_PIN 34 // Pin analógico del sensor de humedad
26 #define HUMIDITY_THRESHOLD 45 // Umbral de humedad mínima (en %)
27
28 bool isWatering = false; // Estado de la bomba
29 unsigned long wateringStartTime = 0; // Tiempo de inicio del riego
30
31 void setup() {
32   Serial.begin(115200);
33
34   // ** Inicializar el servomotor **
35   servoMotor.setPeriodHertz(50);
36   servoMotor.attach(SERVO_PIN);
37   servoMotor.write(0); // Inicialmente cerrar la compuerta
38
39   // ** Inicializar el sensor de luminosidad **
40   Wire.begin();
41   if (!lightMeter.begin(BH1750::CONTINUOUS_HIGH_RES_MODE)) {
42     Serial.println("Error al inicializar el sensor BH1750");
43   }
44 }
```

```
Select Board
actuaadores_jot.ino
73 float temperatureC = sensors.getTempCByIndex(0);
74 if (temperatureC != DEVICE_DISCONNECTED_C) {
75   Serial.print("Temperatura: ");
76   Serial.print(temperatureC);
77   Serial.println(" °C");
78   if (temperatureC > TEMPERATURE_THRESHOLD) {
79     Serial.println("Temperatura alta, encendiendo ventilador...");
80     digitalWrite(FAN_PIN, HIGH);
81   } else {
82     Serial.println("Temperatura normal, apagando ventilador...");
83     digitalWrite(FAN_PIN, LOW);
84   }
85 } else {
86   Serial.println("Error: Sensor de temperatura desconectado");
87 }
88
89 // ** Leer humedad del suelo y controlar la bomba de agua **
90 int rawMoisture = analogRead(SOIL_MOISTURE_PIN);
91 int moistureLevel = map(rawMoisture, 3000, 1000, 0, 100);
92 moistureLevel = constrain(moistureLevel, 0, 100);
93
94 Serial.print("Humedad del suelo: ");
95 Serial.print(moistureLevel);
96 Serial.println("%");
97
98 if (!isWatering) {
99   if (moistureLevel < HUMIDITY_THRESHOLD) {
100     Serial.println("Humedad baja, iniciando riego...");
101     digitalWrite(IN1_PIN, HIGH);
102     digitalWrite(IN2_PIN, LOW);
103     isWatering = true;
104     wateringStartTime = millis();
105   }
106 } else {
107   if (millis() - wateringStartTime >= 2000) {
108     Serial.println("Riego completado, apagando bomba...");
109     digitalWrite(IN1_PIN, LOW);
110     digitalWrite(IN2_PIN, LOW);
111     isWatering = false;
112   }
113 }
114
115 delay(1000); // Intervalo entre cada lectura
```

## Implementación del código

El código presentado integra el control de actuadores y el envío de datos a Azure IoT Central. Está diseñado para gestionar un sistema automatizado que utiliza sensores para monitorear el ambiente y actuadores para responder según parámetros definidos:

- Temperatura: 15 C° - 25 C°
- Humedad: 50% - 70%.
- Luz solar: 10,000 – 25, 000 lux

Además, envía telemetría a la nube para supervisión remota. Sus principales objetivos son controlar los actuadores (ventilador, bomba de agua y servomotor) y enviar los datos recopilados en formato JSON a Azure.

### Los sensores y actuadores son controlados mediante las siguientes variables en el código:

SOIL\_MOISTURE\_PIN: Pin analógico conectado al sensor de humedad del suelo.

FAN\_PIN: Pin digital que controla el ventilador.

IN1\_PIN y IN2\_PIN: Pines digitales que controlan la bomba de agua.

SERVO\_PIN: Pin digital conectado al servomotor.

LIGHT\_THRESHOLD: Umbral de luminosidad para abrir o cerrar la compuerta, en nuestro caso para hacer la demostración usamos 2000, ya que el valor de 25 000 luxes es muy alto

TEMPERATURE\_THRESHOLD: Umbral de temperatura para encender el ventilador, en nuestro caso 25 C°.

HUMIDITY\_THRESHOLD: Umbral de humedad del suelo para activar la bomba de agua en nuestro caso 50%

### Variables de los actuadores:

isWatering: Indica si la bomba de agua está activa.

wateringStartTime: Registra el tiempo de inicio del riego para calcular la duración.

### Telemetría

data\_buffer: Búfer donde se construye el JSON de telemetría.

azure\_iot: Estructura que gestiona la conexión y publicación de datos en Azure IoT.

### Funciones Principales

loop: Maneja los actuadores en función de los datos de los sensores, y se encarga de llamar a las funciones para enviar la telemetría.

## Generación y Envío de Telemetría

generate\_telemetry\_payload: Crea un JSON con los datos de temperatura, humedad y luminosidad.

### 6. Infraestructura Física:

- Diseñamos e imprimimos en MDF una base para colocar la maceta junto con los sensores y actuadores, garantizando un montaje estable y funcional.

## Componentes físicos:

- Sensor de humedad Sensor capacitivo de humedad del suelo V2.0: **150 – 200 mxn**
- Sensor de temperatura DS18B20: **80 - 100 mxn**
- Sensor de luminosidad BH1750: **100 – 150 mxn**
- Protoboards, placa de desarrollo, material de cableado: **150 – 300 mxn**
- Software:
  - IDE de Arduino: **Gratuito**
  - Base de datos en la nube (Azure) 30 días: **Gratuito.**
- Extras: ·
  - Brote de planta\*: **100 mxn** \*Dependiendo del tipo del brote y tamaño de la maceta.
  - placas de MDF de 1.22\* 61 CM: **180 mxn**
- TOTAL, ESTIMADO: **700 – 1500 MXN**
- LIMITE DE GASTOS: **2000 MXN**

## En caso de haber comprado actuadores

- Mini Bomba de Agua 3V-6V Sumergible: **80 – 120 mxn**
- Servomotor SG90 RC 9g: **40 – 80 mxn**
- ventilador 5V para Raspberry Pi (30x30mm): **60 – 120 mxn**

## Azure IoT

Azure IoT (Internet of Things) es un servicio de la nube de Microsoft diseñado para conectar, monitorear y administrar dispositivos IoT de manera segura. Ofrece una infraestructura que permite a dispositivos como la ESP32 enviar datos, recibir comandos y gestionar propiedades, facilitando el monitoreo y control de sistemas remotos.

### ¿Qué es el Azure IoT Hub?

El **Azure IoT Hub** es la pieza central del ecosistema IoT en Azure. Funciona como un intermediario entre dispositivos físicos (como sensores y actuadores conectados a una ESP32) y las aplicaciones en la nube. Sus principales funciones son:

- **Recibir datos de los dispositivos** (telemetría).
- **Administrar propiedades** del dispositivo, como actualizaciones de configuración.
- **Asegurar las comunicaciones** mediante protocolos seguros como MQTT.

### IoT Dashboard

El **IoT Dashboard** es una interfaz gráfica que permite visualizar en tiempo real los datos enviados por los dispositivos conectados al IoT Hub. Este dashboard se utiliza para:

- Monitorear las métricas más recientes enviadas por los sensores.
- Analizar tendencias y comportamientos de las variables.

-

En nuestro proyecto, el IoT Dashboard muestra los datos de telemetría más recientes de los sensores (temperatura, humedad del suelo, luminosidad), lo que permite monitorear el entorno y tomar decisiones en tiempo real.

## Conexión entre la ESP32 y Azure IoT Hub

### 1. Registro del Dispositivo en Azure IoT Hub:

- Primero, se registra la ESP32 como un dispositivo en Azure IoT Hub

### 2. Configuración de la ESP32:

- Se utiliza el SDK de Azure IoT para configurar la conexión. Esto incluye:
  - Proveer las credenciales del dispositivo.
  - Establecer el protocolo de comunicación (usamos **MQTT** en este caso).
- La ESP32 establece una conexión segura con Azure IoT Hub.

### 3. Lectura y Envío de Datos:

- Los sensores conectados a la ESP32 miden variables como temperatura, humedad del suelo y luminosidad.
- Los datos se formatean como un objeto JSON usando la función `generate_telemetry_payload`.

```
static int generate_telemetry_payload(
    uint8_t* payload_buffer,
    size_t payload_buffer_size,
    size_t* payload_buffer_length);
static int generate_device_info_payload(
    const az_iot_hub_client* hub_client,
    uint8_t* payload_buffer,
    size_t payload_buffer_size,
    size_t* payload_buffer_length) {
    az_json_writer jw;
    az_result rc;
    az_span payload_buffer_span = az_span_create(payload_buffer, payload_buffer_size);

    Serial.println("Iniciando generación de payload...");

    rc = az_json_writer_init(&jw, payload_buffer_span, NULL);
    if (az_result_failed(rc)) {
        Serial.println("Error al inicializar el escritor JSON.");
        return RESULT_ERROR;
    }

    rc = az_json_writer_append_begin_object(&jw);
    if (az_result_failed(rc)) {
        Serial.println("Error al abrir el objeto JSON.");
        return RESULT_ERROR;
    }
}
```

- Estos datos JSON se envían al IoT Hub mediante la función `azure_pnp_send_telemetry`.

```
int azure_pnp_send_telemetry(azure_iot_t* azure_iot) {
    Serial.println("Entrando a azure_pnp_send_telemetry...");
    int result;
    size_t payload_length;

    result = generate_telemetry_payload(data_buffer, DATA_BUFFER_SIZE, &payload_length);
    if (result != RESULT_OK) {
        Serial.println("Error generando payload de telemetría.");
        return RESULT_ERROR;
    }
    Serial.println("Payload de telemetría generado con éxito:");

    // Imprime el payload generado (para verificar qué se enviará)
    Serial.println((char*)data_buffer);

    sendJSONToPHP(data_buffer, payload_length);

    result = azure_iot_send_telemetry(azure_iot, az_span_create(data_buffer, payload_length));
    if (result != 0) {
        Serial.println("Error enviando telemetría.");
        return RESULT_ERROR;
    }
    Serial.println("Telemetría enviada con éxito.");
    return RESULT_OK;
}
```

#### 4. Visualización en el IoT Dashboard:

- Los datos enviados al IoT Hub se procesan y se muestran automáticamente en el dashboard.
- Cada vez que la ESP32 envía nuevos datos, el dashboard se actualiza con las lecturas más recientes.



## Resultados

Los resultados obtenidos fueron los esperados debido a que logramos concluir la elaboración de nuestra maceta inteligente dentro del tiempo establecido.

Este éxito no solo se debe a la buena planificación, sino también al esfuerzo colaborativo y al enfoque que aplicamos en cada fase del proyecto. Desde la selección de los componentes adecuados, como los sensores de temperatura, humedad del suelo y radiación solar, hasta la integración eficiente con el sistema IoT, todo fue llevado a cabo con una visión clara y metas bien definidas desde el principio de la materia de Implementación del internet de las cosas.

Fuimos capaces de monitorear las variables críticas de la planta de menta de manera precisa y constante, lo que permitió ajustar el sistema a las necesidades específicas del cultivo. La implementación de sensores nos permitió obtener datos en tiempo real, los cuales fueron procesados por el microcontrolador ESP32 lo que facilitó la toma de decisiones automáticas como la activación del riego y la apertura de la puerta para la luz solar

## Conclusiones

- **Integración Tecnológica Exitosa**

El proyecto logró implementar una conexión funcional entre la ESP32 y Azure IoT Hub, habilitando la captura, transmisión y visualización en tiempo real de variables clave como temperatura, luminosidad y humedad del suelo. Esto demuestra la capacidad del sistema para gestionar datos de telemetría y ofrecer soluciones tecnológicas avanzadas.

- **Monitoreo Centralizado y Eficiente**

La integración con el IoT Dashboard de Azure permitió centralizar la supervisión de las condiciones del entorno de manera accesible y en tiempo

real. Este avance garantiza una gestión eficiente de los datos, lo que facilita la toma de decisiones basada en información actualizada.

- **Diseño Modular y Escalable**

La arquitectura del sistema fue diseñada para ser flexible y escalable, permitiendo la integración de nuevos sensores, actuadores o funcionalidades sin requerir cambios significativos en su infraestructura inicial.

- **Automatización de Procesos**

La implementación de lógica para el control de actuadores, como bombas de agua y LEDs, demostró la capacidad del sistema para responder automáticamente a cambios en las variables monitoreadas. Esto optimiza el uso de recursos y reduce la necesidad de intervención manual.

- **Infraestructura Física Funcional**

La construcción de una base en MDF permitió consolidar una integración práctica y estable entre los sensores y actuadores. Este diseño asegura que el sistema pueda operar de manera confiable en entornos reales.

Durante el desarrollo, se superaron retos como la complejidad del código y problemas iniciales de conexión con Azure IoT Hub. Estos obstáculos fueron resueltos de manera efectiva, demostrando la capacidad del equipo para abordar y superar desafíos técnicos.

El sistema desarrollado tiene un alto potencial para aplicaciones reales, como la automatización agrícola o el monitoreo ambiental. Además, abre la posibilidad de incorporar análisis de datos históricos mediante bases de datos y personalizar las visualizaciones para usuarios finales.