Sistema de Monitoreo para Huertos con Alertas Automatizadas

Jesús Rodríguez Sánchez

Introducción

Este proyecto tiene como objetivo desarrollar un sistema de monitoreo para huertos que permita a los usuarios observar las condiciones ambientales de sus cultivos desde cualquier lugar del mundo. A través del uso de sensores conectados a una Raspberry Pi, y con integración web y notificaciones automáticas, se facilita la gestión de los cultivos.

Descripción General del Proyecto

Este sistema monitorea variables ambientales clave como la **humedad del suelo**, la **temperatura** y la **luz** mediante sensores conectados a una **Raspberry Pi**. Un **script en Python** recoge y envía los datos a una **base de datos TimescaleDB** alojada en una instancia **EC2 en AWS**, utilizando **Docker** para su despliegue.

Los usuarios acceden a los datos a través de una **aplicación web responsiva** construida con **Bootstrap y Chart.js**, también contenida en Docker. Cuando los valores superan ciertos umbrales, el sistema envía **notificaciones automáticas por Telegram** gracias a la integración con un bot Python.

Además, se realiza una **copia de seguridad diaria automatizada** mediante **cron** y envío por **SFTP**, garantizando la integridad de los datos.

Objetivos del Sistema

- Recopilar y almacenar lecturas ambientales del huerto.
- Visualizar los datos de forma clara y gráfica desde cualquier dispositivo.
- Enviar notificaciones si se detectan condiciones críticas.
- Automatizar copias de seguridad hacia un servidor FTP.
- Favorecer prácticas de agricultura sostenible mediante tecnología.

Requisitos del Sistema

Hardware

- Raspberry Pi (con Wi-Fi)
- Sensor de humedad (digital GPIO)
- Sensor de temperatura DHT22
- Sensor de luz LM393
- Fuente de alimentación estable
- Conexión a Internet

Software

- Raspberry Pi OS / Debian
- Python 3 con bibliotecas**: adafruit_dht, RPi.GPIO**
- Flask + HTML + Bootstrap + Chart.js
- TimescaleDB
- Terraform
- Docker
- Telegram Bot API

Instalación y Configuración

Instalar Raspberry Pi OS.

Descarga: https://www.raspberrypi.com/software/

Se recomienda la **versión de 32 bits (Lite)** por mejor compatibilidad con bibliotecas como RPi.GPIO y adafruit-circuitpython-dht

Instalar dependencias:

sudo apt update

sudo apt install -y python3-pip python3-venv libgpiod2 libffi-dev build-essential libpq-dev

2. Crear y activar entorno virtual

python3 -m venv ~/venv
source ~/venv/bin/activate

3. Requisitos de Python

pip install psycopg2-binary adafruit-circuitpython-dht RPi.GPIO

4. Editar el crontab

crontab -e

***/5 * * * * /bin/bash -c 'source /home/[usuario]/venv/bin/activate && python /home/[usuario]/script/carga.py >> /home/[usuario]/script/log.txt 2>&1'**

Infraestructura en la nube (AWS)

Despliegue con Terraform

Este proyecto utiliza **Terraform** para crear y gestionar la infraestructura en AWS, incluyendo:

- Instancia EC2 con Docker para alojar la base de datos TimescaleDB.
- Grupos de seguridad.
- Subredes y VPC (si no existen ya).
- Elastic IP.
- Automatización de backups.

1. Requisitos previos

- Cuenta de AWS con claves de acceso (Access Key ID y Secret Key).
- AWS CLI configurado (aws configure).
- Terraform instalado:
    - **sudo apt update**
    - **sudo apt install -y unzip**
    - **wget https://releases.hashicorp.com/terraform/1.8.5/terraform_1.8.5_linux_amd64.zip**
    - **unzip terraform_1.8.5_linux_amd64.zip**

- **sudo mv terraform /usr/local/bin/**
- **terraform -v**

Despliegue por Módulos (Terraform)

El proyecto está organizado en módulos independientes de Terraform ubicados en subdirectorios. Para desplegar correctamente toda la infraestructura, debes seguir el siguiente **orden de ejecución**:

Orden de ejecución

1. **Redes (VPC/Subred)**
   Crea la red base que será usada por el resto de servicios.
   a. cd terraform/redes
   b. terraform init
   c. terraform apply
2. **TimescaleDB (Base de datos en EC2)**
   a. cd ../instancias/timescaledb
   b. terraform init
   c. terraform apply
3. **FTP (Servidor de backups)**
   a. cd ../ftp
   b. terraform init
   c. terraform apply
4. **Web (Interfaz visual de monitoreo)**
   a. cd ../web
   b. terraform init
   c. terraform apply

Nota: Puedes destruir cada módulo con **terraform destroy** en su respectivo directorio si necesitas limpiar recursos por separado.

Automatización en la instancia EC2 (Backups diarios)

1. Asignar permisos de ejecución al script

Una vez que el script **backup_script.sh** se haya copiado en la instancia (esto ya lo hace Terraform), asegúrate de que tiene permisos de ejecución:

chmod +x /home/ubuntu/backup_script.sh

Además, verifica que la clave del usuario SFTP tiene permisos adecuados:

chmod 600 /home/ubuntu/.ssh/sftp-user-key.pem

2. Añadir la tarea al crontab
Edita el crontab del usuario ubuntu:

crontab -e

Y añade la siguiente línea al final:

0 9 * * * /home/ubuntu/backup_script.sh >> /home/ubuntu/backup_log.txt 2>&1

Instrucciones para ejecutar alertas.py con entorno virtual y cron

1. Instalación de dependencias

sudo apt update

sudo apt install python3-venv -y

Paso 2: Crear y activar entorno virtual

cd ~/web-content/scripts

python3 -m venv venv_alertas

source venv_alertas/bin/activate

Instalar dependencias del script
pip install psycopg2-binary requests

Agregar al crontab

Editar el crontab del usuario Ubuntu:

crontab -e

Agregar la siguiente línea (por ejemplo, cada 5 minutos):

***/5 * * * * /home/ubuntu/web-content/scripts/venv_alertas/bin/python /home/ubuntu/web-content/scripts/alertas.py >> /home/ubuntu/web-content/scripts/alertas_cron.log 2>&1**

Conexión de sensores a los GPIO de la Raspberry Pi

Material necesario

- **Raspberry Pi** (cualquier modelo con GPIO)
- **Cables hembra-hembra o hembra-macho** (según el sensor)
- **Sensor**

Consideraciones antes de conectar
Consulta la hoja técnica del sensor para no quemarlo con voltajes incorrectos.

Pines rojos: Alimentación. Hay dos pines de 5V (2 y 4) y dos de 3.3V (1 y 17).

Pines negros: GND (tierra), necesarios para cerrar el circuito.

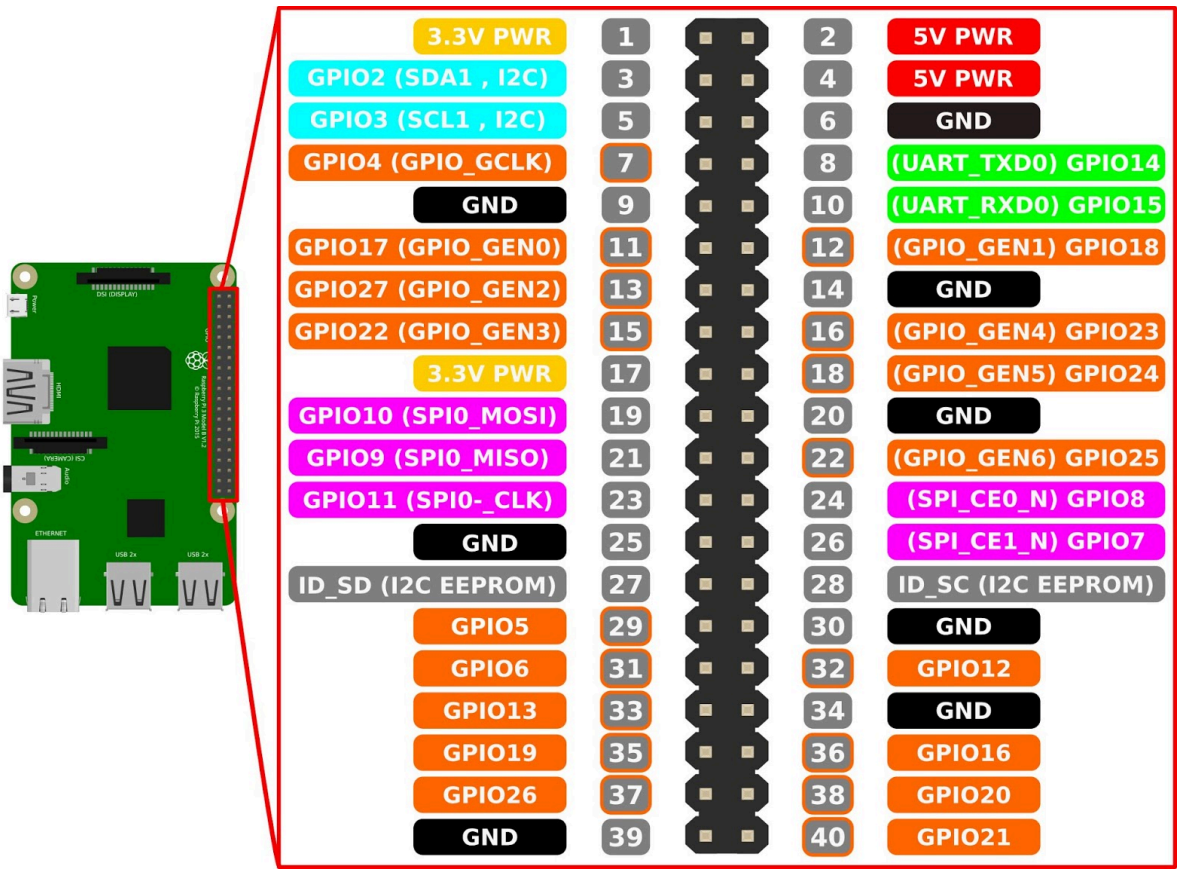
Pines naranjas: GPIO disponibles para entrada o salida digital.

Pines morados y rosados: Pines específicos para interfaces SPI.

Pines verdes: Pines UART (comunicación serie).

Pines celestes: Pines I2C (comunicación con sensores como el DHT o pantallas LCD).

Grises: Pines reservados o para funciones específicas (no usar para sensores comunes).



Configuración dinámica de sensores (sensor_conf.json)
El script principal no necesita estar codificado específicamente para cada sensor. En su lugar, se apoya en un archivo llamado **sensor_conf.json**, que define todos los sensores conectados, sus

tipos y a qué pin GPIO están conectados.
El **script lee** este archivo **JSON** y recorre cada sensor definido.

Según el tipo **(temperatura, humedad, luz)**, llama a la función correspondiente que lee el sensor desde el **GPIO indicado**.
Este diseño hace que el sistema sea modular y fácil de mantener:

Para agregar un **nuevo sensor**, **solo** necesitas **editar el JSON** añadiendo nuevas lineas con sus datos correspondientes.

Uso Normal de la Web

La interfaz web está diseñada para que el usuario final pueda **gestionar su cuenta** y **recibir alertas personalizadas desde Telegram**, sin necesidad de interactuar directamente con la Raspberry Pi.

Flujo típico de uso:

1. **Inicio de sesión**
   - El usuario accede a la web y se autentica con sus credenciales.
   - Solo los usuarios registrados pueden entrar y gestionar su configuración.
2. **Configuración del Chat ID de Telegram**
   - Una vez dentro, accede a la sección de configuración **(chat_id.php).**
   - Aquí puede ingresar o actualizar su **chat_id_telegram**, que es el identificador único que el bot de Telegram necesita para enviarle mensajes.
   - Este ID se guarda en la base de datos y se vincula con su cuenta.
3. **Recepción de Alertas**
   - El script Python en la Raspberry Pi detecta las lecturas de los sensores.
   - Si una alerta está activa y las condiciones se cumplen, el script consulta la base de datos para obtener el **chat_id** del usuario.
   - Entonces, le envía un mensaje por Telegram notificando el estado del sensor (por ejemplo: temperatura alta, falta de luz, etc.).

4. **Dashboard o navegación adicional**
   - Desde la web el usuario puede volver al panel principal, cerrar sesión, y eventualmente consultar información adicional (si se agregan más páginas como historial, gráficas, etc.).

Tecnologías Utilizadas

- **Hardware**: Raspberry Pi, DHT22, LM393, sensores digitales
- **Lenguajes**: Python, JavaScript, HTML, Terraform
- **Frameworks/Librerías**: Flask, Bootstrap, Chart.js
- **Base de Datos**: PostgreSQL con TimescaleDB
- **Orquestación/Infraestructura**: Docker, Terraform, AWS EC2
- **Notificaciones**: Bot de Telegram
- **Backups**: Crontab + SFTP

Funcionamiento del Sistema

Flujo de Datos

1. La Raspberry Pi toma una lectura de los sensores.
2. Los datos se envían al servidor (EC2) a través de una API.
3. El backend almacena los datos en TimescaleDB.
4. Si los valores superan umbrales definidos, se crea una alerta y se envía una notificación por Telegram.
5. El frontend consulta y visualiza las lecturas mediante Chart.js.