Avance del Proyecto de Ambientes Inteligentes

Sistema de Riego Automatizado Inteligente con Notificaciones vía Telegram

PAO I Término 2024 - 2025

Milton Josué Sánchez Véliz

David Alejandro Romero Yánez

Rafael Johan Estrada Rodríguez

Escuela Superior Politécnica del Litoral

Facultad de Ingeniería Mecánica y Ciencias de la Producción (FIMCP)

Ingeniería en Mecatrónica

Guayaquil – Ecuador

miljosan@espol.edu.ec – daroyane@espol.edu.ec – rjestrad@espol.edu.ec

OBJETIVOS

- Automatizar el riego de cultivos basándose en parámetros de temperatura, humedad y horario.
- Permitir la supervisión y control remoto del sistema de riego mediante Telegram.
- Optimizar el uso de agua en los cultivos para evitar riegos excesivos o insuficientes.

1. RESUMEN

El proyecto consiste en el desarrollo de un sistema de riego automatizado que utiliza sensores de temperatura y humedad para determinar las necesidades de riego de un cultivo. Este sistema enviará notificaciones al usuario a través de la aplicación de mensajería Telegram, permitiendo además el control remoto del riego. La automatización del riego se basará en datos en tiempo real de las condiciones ambientales y horarios programados. Este proyecto busca mejorar la eficiencia del riego y asegurar la salud óptima de los cultivos mediante el uso de tecnologías avanzadas.

2. INTRODUCCIÓN

La automatización en la agricultura es una tendencia creciente que busca optimizar los procesos y reducir el esfuerzo manual, promoviendo una mayor eficiencia y sostenibilidad. En el ámbito del riego, la automatización puede transformar significativamente la forma en que se gestiona este recurso crítico, asegurando que los cultivos reciban la cantidad adecuada de agua en el momento justo. Este proyecto se centra en el desarrollo de un sistema de riego automatizado que utiliza sensores de

temperatura y humedad para monitorear continuamente las condiciones ambientales y determinar las necesidades de riego de un cultivo de manera precisa.

El sistema propuesto integra la aplicación de mensajería Telegram para permitir una interacción remota y dinámica con el usuario. A través de Telegram, el sistema enviará notificaciones sobre el estado del riego y las condiciones del suelo, y permitirá al usuario activar o desactivar el riego manualmente si lo desea. Esto no solo proporciona una mayor flexibilidad y control, sino que también reduce la necesidad de intervención manual constante, permitiendo a los agricultores concentrarse en otras tareas críticas.

Además, la automatización del riego basada en datos en tiempo real y horarios programados puede mejorar significativamente la eficiencia del uso del agua, reduciendo el desperdicio y asegurando que los cultivos reciban el riego adecuado. Esto es particularmente importante en áreas con recursos hídricos limitados o en situaciones donde el riego preciso es crucial para la salud y el rendimiento de los cultivos.

La implementación de estas tecnologías no solo facilita la creación de un sistema de riego inteligente, sino que también proporciona una base sólida para futuras expansiones y mejoras. Al combinar sensores avanzados, automatización, y comunicación remota, este proyecto ofrece una solución integral para la gestión eficiente del riego en la agricultura moderna.

3. JUSTIFICACIÓN

En la agricultura moderna, la gestión eficiente del agua es un desafío crucial, especialmente en un contexto global marcado por la escasez de recursos hídricos y el cambio climático. Las prácticas tradicionales de riego, que a menudo dependen de la intervención manual y la experiencia del agricultor, no siempre garantizan el uso óptimo del agua, lo que puede llevar a desperdicios y a una menor productividad de los cultivos.

El desarrollo del sistema de riego automatizado propuesto en este proyecto responde a la necesidad de modernizar estas prácticas mediante una solución que combina precisión, eficiencia y accesibilidad económica. Utilizando sensores de temperatura y humedad, este sistema será capaz de monitorear en tiempo real las condiciones del suelo y tomar decisiones informadas sobre el riego, asegurando que los cultivos reciban la cantidad justa de agua en el momento adecuado. Esta tecnología no solo mejora la productividad agrícola, sino que también contribuye a la conservación del agua, un recurso cada vez más valioso.

Un aspecto clave de este proyecto es su enfoque en la accesibilidad y el bajo costo, lo que lo convierte en una solución ideal tanto para estudiantes de agricultura como para empresas que están comenzando en este campo. Al ofrecer una tecnología avanzada y asequible, el sistema permite que incluso pequeños agricultores o empresas emergentes puedan beneficiarse de la automatización, mejorando la sostenibilidad de sus operaciones sin incurrir en gastos prohibitivos. Además, la integración con la aplicación de mensajería Telegram agrega un valor significativo al permitir una gestión remota y flexible del riego, lo que reduce la necesidad de intervención manual constante y ofrece un control más preciso sobre los cultivos.

En resumen, este proyecto no solo se justifica por su capacidad para mejorar la eficiencia del riego y conservar recursos vitales, sino también por su diseño pensado para ser una solución económicamente accesible y de alto valor. Al facilitar la adopción de tecnologías avanzadas en un formato asequible, este sistema tiene el potencial de transformar tanto el aprendizaje de estudiantes en el campo de la agricultura como la eficiencia de empresas que inician su camino en este sector, contribuyendo a un modelo agrícola más sostenible y productivo.

4. DISEÑO DE LA ARQUITECTURA TECNOLÓGICA

Figure 1. Arquitectura Tecnológica

5. MATERIALES A USAR Y SU RESPECTIVA DESCRIPCIÓN

1. Microcontrolador (ESP32)

Es el cerebro del sistema, encargado de procesar los datos recibidos de los sensores y de controlar las válvulas de riego y otros actuadores. El ESP32, en particular, incluye conectividad Wi-Fi y Bluetooth, lo que facilita la integración con la aplicación de mensajería Telegram.

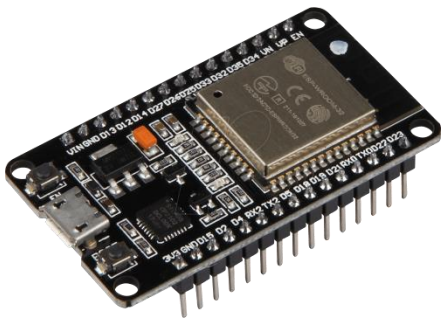

Figure 1. ESP32

2. Sensores de Humedad del Suelo

Estos sensores miden la cantidad de agua presente en el suelo. El microcontrolador utiliza esta información para determinar si el suelo necesita riego.

Figure 3. Sensor de humedad del suelo YL38 y MO750

3. Sensores de Temperatura y Humedad del Aire

Miden la temperatura y la humedad del ambiente. Estos datos pueden utilizarse para ajustar el riego según las condiciones climáticas actuales.

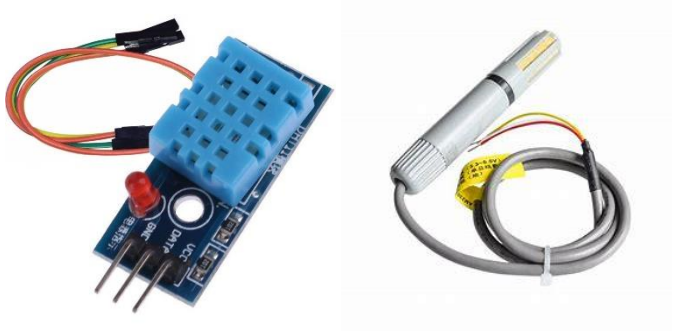

Figure 4. Sensores de Temperatura y Humedad Dht11 y AM2305B

4. Válvulas de Riego Electrónicas

Son dispositivos que controlan el flujo de agua hacia los cultivos. Están conectadas al microcontrolador, que las abre o cierra en función de los datos proporcionados por los sensores.

Figure 5. Electroválvula riego cepex 9V LATCH

5. Bomba de Agua

Si el sistema de riego requiere presión adicional, una bomba de agua es necesaria para suministrar la cantidad adecuada de agua a los cultivos.

Figure 6. Bomba de agua eléctrica periférica

6. Conectores y Cables

Son necesarios para conectar todos los componentes del sistema, como los sensores, las válvulas, la bomba de agua y el microcontrolador.

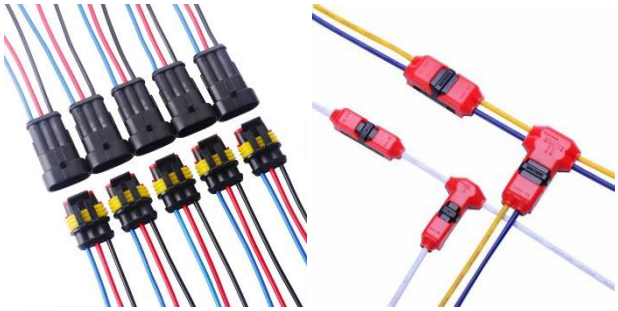

Figure 7. Conectores de cables eléctricos

7. Fuente de Alimentación (Adaptador de corriente o Baterías)

Provee la energía necesaria para que funcione el microcontrolador y otros componentes electrónicos. Las baterías recargables o paneles solares también pueden ser opciones, especialmente en zonas rurales.

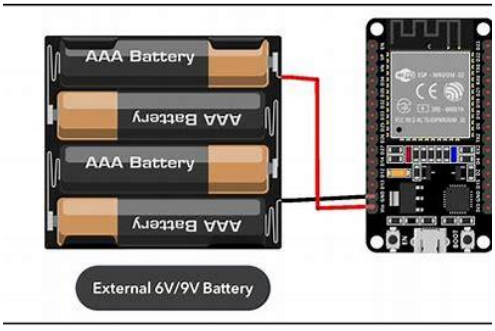

Figure 8. Pilas para alimentar a la ESP32

8. Módulo Wi-Fi (Integrado en ESP32)

Permite la conectividad del sistema a Internet, esencial para enviar y recibir datos a través de la aplicación Telegram.

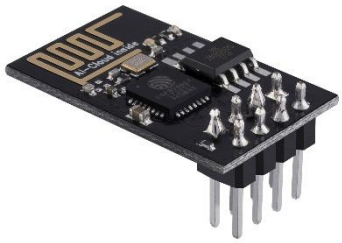

Figure 9. Módulo Wi-Fi para ESP32

9. Tuberías de Riego

Descripción: Conducen el agua desde la fuente hasta los cultivos. Dependiendo del tamaño y las necesidades del sistema, pueden variar en tamaño y material.

Figure 10. Tuberías para Sistema de riego

10. Tanque de Almacenamiento de Agua (Opcional)

Si no hay una fuente de agua constante, un tanque puede almacenar el agua necesaria para el riego.

Figure 11. Tanque para el almacenamiento de agua

11. Módulo Relé

Es utilizado para controlar dispositivos de mayor potencia, como la bomba de agua o las válvulas, con señales de baja potencia provenientes del microcontrolador.

Figure 12. Tanque para el almacenamiento de agua

12. Módulo RTC (Real Time Clock)

Un reloj en tiempo real que permite programar el riego en momentos específicos del día, independientemente de la conexión a Internet.

Figure 13. DS3231 Módulo RTC Reloj tiempo real

13. Conectores y Uniones de Tubería

Componentes para unir diferentes tramos de tuberías, asegurando que el sistema de riego cubra todas las áreas necesarias.

Figure 14. Conectores de tuberías

14. Enclosure o Caja Protectora

Caja resistente a la intemperie para proteger el microcontrolador y otros componentes electrónicos de las condiciones climáticas.

Figure 15. Caja protectora de componentes eléctricos

15. Software de Programación (Arduino IDE o Plataforma similar)

Herramienta necesaria para programar el microcontrolador y definir el comportamiento del sistema de riego.

Figure 16. Arduino

6. METODOLOGÍA

a. Diseño

Este proyecto se diseñó como un estudio experimental y educativo, cuyo objetivo principal es proporcionar a los estudiantes de agricultura una experiencia práctica en la implementación y evaluación de tecnologías de automatización, específicamente en sistemas de riego. Se busca no solo instruir sobre la instalación y programación, sino también fomentar el análisis crítico y comparativo frente a métodos tradicionales.

b. Población

La población de este estudio está compuesta por un grupo específico de **estudiantes de agricultura** que están cursando un programa de estudios enfocado en la tecnología aplicada a la agricultura. Estos estudiantes se seleccionan en función de su interés y experiencia previa en proyectos relacionados con la automatización y el manejo de

cultivos. El marco de la muestra incluye estudiantes de distintos niveles, lo que permite una evaluación comparativa de cómo los conocimientos previos influyen en la implementación y éxito del sistema.

c. Entorno

El estudio se llevará a cabo en un **invernadero experimental** perteneciente a la institución educativa donde estudian los participantes. Este entorno controlado permite a los estudiantes experimentar con las variables ambientales de manera segura y obtener datos precisos sobre el rendimiento del sistema de riego. Además, se seleccionará un área de campo abierta en la institución para simular condiciones más cercanas a las del mundo real, proporcionando una experiencia completa y variada.

d. Intervenciones

Las intervenciones en este proyecto incluyen:

- **Instalación de Sensores y Sistema de Control:** Los estudiantes instalarán sensores de humedad y temperatura en las parcelas de cultivo. Estos sensores estarán conectados a un microcontrolador que automatizará el riego en función de los datos recolectados.
- **Programación y Configuración del Sistema:** Los estudiantes aprenderán a programar el microcontrolador para procesar los datos de los sensores y controlar las válvulas de riego. También configurarán la interfaz con la aplicación Telegram para el monitoreo remoto y la intervención manual.
- **Pruebas Piloto:** Se llevarán a cabo pruebas piloto en pequeñas áreas del invernadero para ajustar los parámetros del sistema, como los umbrales de humedad y las horas de riego, antes de su implementación completa en las parcelas de cultivo.
- **Monitoreo del Sistema:** Los estudiantes estarán encargados de monitorear el sistema de riego durante el período del estudio, recopilando datos sobre la eficiencia del uso del agua, el crecimiento de las plantas y las condiciones del suelo.
- **Evaluación Comparativa:** Al finalizar el experimento, los estudiantes compararán los resultados obtenidos en las parcelas con riego automatizado frente a las parcelas con riego manual, evaluando la eficiencia y efectividad del sistema implementado.

7. RESULTADOS

La simulación del proyecto fue realizada en el software Cisco Packet Tracer, cuyas imágenes de su ejecución se pueden observar en la sección de anexos.

Para el encendido y apagado de los aspersores se programó un código en un “Switching Bus Controller 4” (SBC4 por sus siglas en inglés) utilizando el lenguaje de programación Python; en él se definieron umbrales máximos y mínimos para encender o apagar los aspersores dependiendo del caso (Figure 18). Cuando están apagados los

aspersores (Figure 17) es porque los sensores están detectando niveles altos de humedad y/o una temperatura baja, por lo que no hay necesidad de riego. Mientras que los aspersores se encienden cuando se detectan niveles altos de temperatura y/o bajos de humedad (Figure 19)

Finalmente tenemos una representación de como sería un funcionamiento simple de la activación manual de los aspersores y la lectura de los sensores desde Telegram usando un bot. (Figure 20)

Cabe recalcar que cuando los aspersores ejecutan una acción, se envía una señal al ESP32, la cual la transmite al smartphone por medio del Gateway utilizando un módulo WiFi; de esta manera los datos que arrojan los sensores, así como lo que estén realizando los aspersores le llega al usuario de una forma fácil de entender ya que se usa la aplicación Telegram como medio de comunicación entre el usuario y el sistema de riego automatizado.

8. ANÁLISIS DE RESULTADOS

La arquitectura del sistema implementado consta de los siguientes componentes principales: un sensor de temperatura, un sensor de humedad, tres aspersores, un microcontrolador ESP32, un gateway, y un teléfono celular. Todos los elementos están conectados al ESP32, que actúa como el núcleo de control del sistema.

El proceso comienza con la recolección de datos de temperatura y humedad por los sensores. Estos datos se envían al ESP32, que, mediante la programación desarrollada, evalúa los valores en tiempo real. Dependiendo de las condiciones ambientales, como un nivel de humedad bajo o una temperatura elevada que indique la necesidad de riego, el ESP32 toma la decisión de encender o apagar los aspersores.

Un aspecto clave del sistema es su capacidad de retroalimentación. Cada vez que los aspersores se activan o desactivan, el ESP32 envía una notificación instantánea a través de la aplicación Telegram al cliente final. Este mensaje incluye información actualizada sobre la temperatura y la humedad en el momento de la acción, lo que permite al usuario estar informado del estado del riego sin necesidad de supervisión directa.

Resultados observados:

- **Fiabilidad del sistema:** Las simulaciones en Cisco Packet Tracer y en el IDE de Arduino mostraron una alta fiabilidad en la transmisión de datos desde los sensores hacia el ESP32, y en la toma de decisiones automática respecto a la activación de los aspersores.
- **Eficiencia de riego:** El sistema demostró mejorar la eficiencia en el uso del agua, activando los aspersores solo cuando las condiciones lo requerían, reduciendo así el desperdicio de recursos.
- **Notificaciones en tiempo real:** Las pruebas realizadas con Telegram confirmaron que el sistema envía las notificaciones al instante, asegurando que el usuario final esté constantemente informado sobre el estado del sistema de riego.

Este sistema de riego automatizado ha mostrado ser eficaz en la simulación para optimizar tanto el uso de recursos hídricos como la supervisión remota del proceso de riego, mejorando así el control y la sostenibilidad en el manejo de los cultivos.

9. CONCLUSIONES

- El sistema de riego automatizado ha demostrado ser una herramienta eficaz para optimizar el uso de agua en la agricultura. Al activar los aspersores solo cuando las condiciones de temperatura y humedad lo requieren, se reduce significativamente el desperdicio de agua, lo que es especialmente crítico en áreas con recursos hídricos limitados.
- La integración de sensores de temperatura y humedad con el microcontrolador ESP32 ha permitido la automatización precisa del riego. La capacidad del sistema para tomar decisiones en tiempo real garantiza que los cultivos reciban la cantidad adecuada de agua, mejorando así la productividad y la salud de las plantas.
- La implementación de la mensajería vía Telegram ha añadido un nivel de comodidad y control para el usuario final. La capacidad de recibir notificaciones en tiempo real sobre las condiciones del ambiente y las acciones del sistema de riego permite una supervisión efectiva sin la necesidad de intervención directa, lo que facilita la gestión del riego desde cualquier lugar.

10. RECOMENDACIONES

- Considerar la integración de sensores adicionales, como sensores de humedad del suelo o de luz solar. Esto podría proporcionar datos más completos, permitiendo al sistema tomar decisiones de riego aún más precisas, ajustándose mejor a las necesidades específicas de los cultivos.
- Introducir algoritmos de aprendizaje automático podría mejorar la eficiencia del sistema al permitir que se ajuste de manera autónoma a las condiciones cambiantes a lo largo del tiempo. Por ejemplo, el sistema podría aprender patrones estacionales y ajustar automáticamente los parámetros de riego según la experiencia previa.
- Explora la posibilidad de alimentar el sistema con energía solar o de otras fuentes renovables. Esto no solo haría el sistema más sostenible, sino que también reduciría los costos operativos y aumentaría su viabilidad en áreas remotas.
- Aunque Telegram ofrece una manera sencilla de supervisar el sistema, considera el desarrollo de una interfaz gráfica más robusta, como una aplicación móvil dedicada o una plataforma web. Esto podría proporcionar un control más intuitivo y acceso a estadísticas históricas, lo que facilitaría la gestión a largo plazo.

11. ANEXOS

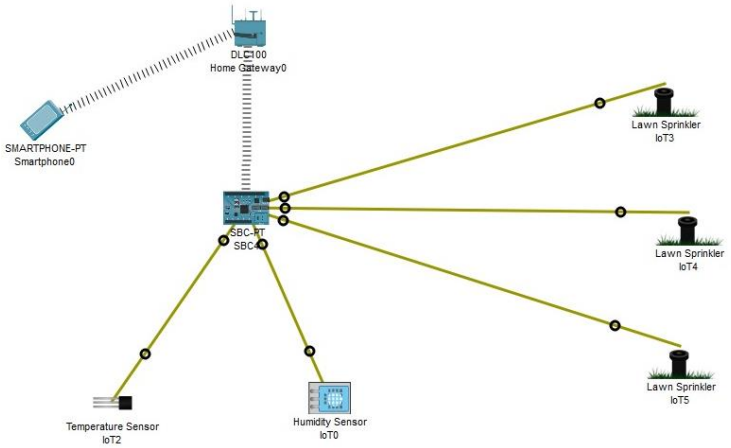

Figure 17. Diagrama de conexiones con aspersores en OFF

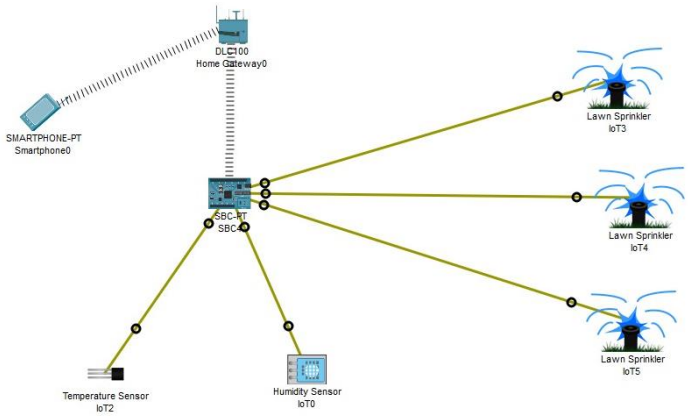

Figure 18. Diagrama esquemático aspersores en ON

SBC4

Specifications | Physical | Config | Desktop | Programming | Attributes

New Project (Python) - main.py

Open | New | Delete | Rename | Import

main.py

```python
from gpio import *
from time import *
import requests

# Configuración de los pines
TEMP_SENSOR_PIN = 0  # Cambiar al pin adecuado
HUM_SENSOR_PIN = 1   # Cambiar al pin adecuado
ASPERSOR_PINS = [2, 3, 4]  # Pines digitales para los aspersores

# Umbrales para activar los aspersores
TEMP_THRESHOLD = 0
HUM_THRESHOLD = 0

# Configuración de Wi-Fi
SERVER_URL = "http://192.168.25.104/update"  # URL del servidor al que enviar los datos

def read_sensor(pin):
    # Esta función debe leer el valor del sensor conectado al pin especificado
    # Implementa la lectura del sensor aquí (esto depende del sensor y la biblioteca que uses)
    return 0  # Cambiar por la lectura real del sensor

def setup():
    # Inicializar los pines
    for pin in ASPERSOR_PINS:
        pinMode(pin, OUT)

def control_aspersores(temp, hum):
    # Controlar los aspersores según los umbrales
    if temp < TEMP_THRESHOLD and hum < HUM_THRESHOLD:
        for pin in ASPERSOR_PINS:
            customWrite(pin, 1)
        print("Prendiendo aspersores")
    else:
        for pin in ASPERSOR_PINS:
            customWrite(pin, 0)
        print("Apagando aspersores")

def send_data(temp, hum):
    # Enviar datos al servidor
    payload = {'temp': temp, 'hum': hum}
    try:
        response = requests.get(SERVER_URL, params=payload)
        print("Datos enviados:", response.status_code)
    except Exception as e:
        print("Error al enviar datos:", e)

def main():
    setup()
    print("Enviando datos...")
    while True:
        temp = read_sensor(TEMP_SENSOR_PIN)
        hum = read_sensor(HUM_SENSOR_PIN)

        control_aspersores(temp, hum)
```

Figure 19. Código para funcionamiento del bot en Telegram

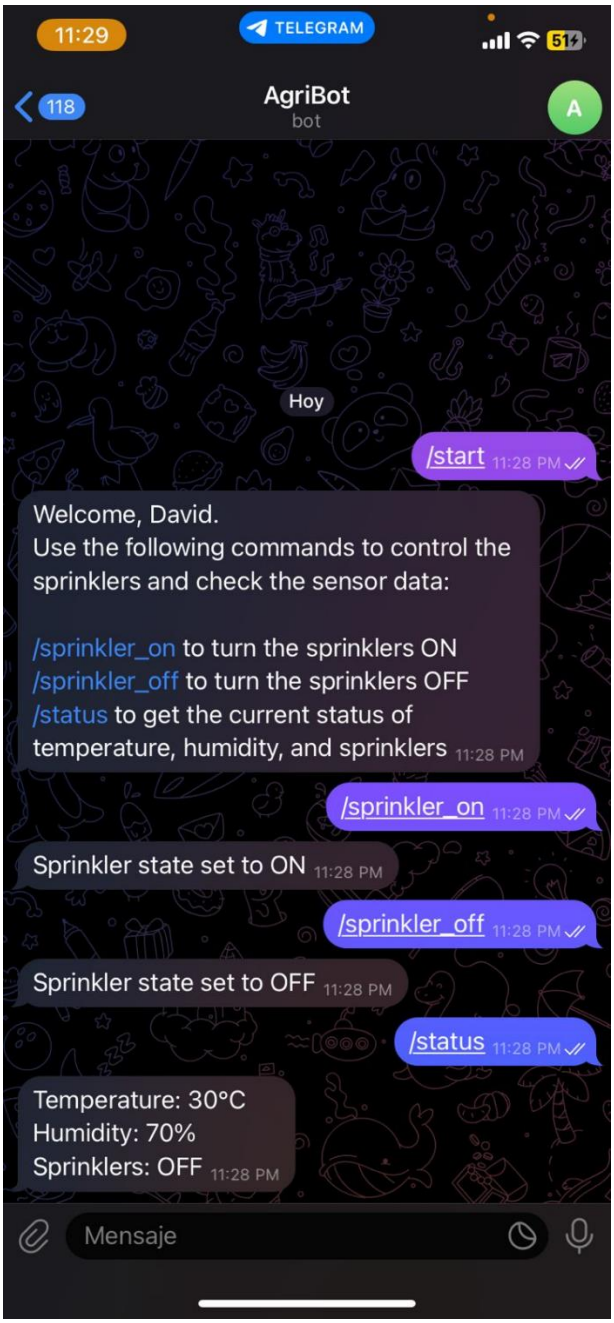

Figure 20. Bot de Telegram en funcionamiento

Código de la ESP32 para el funcionamiento del Bot de Telegram y procesamiento de señales:

```
#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <UniversalTelegramBot.h>
```

```
#include <ArduinoJson.h>
#include <DHT.h>

// Wifi network station credentials
#define WIFI_SSID "Lab. Telematica"
#define WIFI_PASSWORD "l4bt3l3m4tic@"
// Telegram BOT Token (Get from Botfather)
#define BOT_TOKEN "7331988786:AAEnQ-oLcUrXCwxBiFnm1g691t1ECV6o6sw"

// Use @myidbot (IDBot) to find out the chat ID of an individual or a group
#define CHAT_ID "175753388"

// DHT Sensor
#define DHTPIN 0
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);

// Sprinkler control
#define SPRINKLER_PIN 5    // Pin connected to the relay or control module

WiFiClientSecure secured_client;
UniversalTelegramBot bot(BOT_TOKEN, secured_client);

// Checks for new messages every 1 second.
int botRequestDelay = 1000;
unsigned long lastTimeBotRan;

void setup() {
  Serial.begin(115200);
  Serial.println();

  // Initialize DHT sensor
```

```
  dht.begin();

  // Initialize Sprinkler Pin
  pinMode(SPRINKLER_PIN, OUTPUT);
  digitalWrite(SPRINKLER_PIN, LOW); // Assume LOW means off

  // Attempt to connect to Wifi network
  Serial.print("Connecting to Wifi SSID ");
  Serial.print(WIFI_SSID);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  secured_client.setCACert(TELEGRAM_CERTIFICATE_ROOT);  // Add root certificate for api.telegram.org
  while (WiFi.status() != WL_CONNECTED)
  {
    Serial.print(".");
    delay(500);
  }
  Serial.print("\nWiFi connected. IP address: ");
  Serial.println(WiFi.localIP());

  Serial.print("Retrieving time: ");
  configTime(0, 0, "pool.ntp.org"); // Get UTC time via NTP
  time_t now = time(nullptr);
  while (now < 24 * 3600)
  {
    Serial.print(".");
    delay(100);
    now = time(nullptr);
  }
  Serial.println(now);

  // Send a message to Telegram
```

```
  bot.sendMessage(CHAT_ID, "Bot started up", "");
}

void handleNewMessages(int numNewMessages) {
  Serial.println("handleNewMessages");
  Serial.println(String(numNewMessages));

  for (int i = 0; i < numNewMessages; i++) {
    String chat_id = String(bot.messages[i].chat_id);
    if (chat_id != CHAT_ID) {
      bot.sendMessage(chat_id, "Unauthorized user", "");
      continue;
    }

    String text = bot.messages[i].text;
    Serial.println(text);

    if (text == "/start") {
      String welcome = "Welcome! Use the following commands to control the sprinklers and check the sensor data:\n\n";
      welcome += "/sprinkler_on to turn the sprinklers ON\n";
      welcome += "/sprinkler_off to turn the sprinklers OFF\n";
      welcome += "/status to get the current status of temperature, humidity, and sprinklers\n";
      bot.sendMessage(chat_id, welcome, "");
    } else if (text == "/sprinkler_on") {
      bot.sendMessage(chat_id, "Sprinkler state set to ON", "");
      digitalWrite(SPRINKLER_PIN, HIGH);
    } else if (text == "/sprinkler_off") {
      bot.sendMessage(chat_id, "Sprinkler state set to OFF", "");
      digitalWrite(SPRINKLER_PIN, LOW);
    } else if (text == "/status") {
      float temperature = dht.readTemperature();
```

```
      float humidity = dht.readHumidity();

      if (isnan(temperature) || isnan(humidity)) {
        bot.sendMessage(chat_id, "Failed to read from DHT sensor!", "");
        continue;
      }

      String sprinklerState = (digitalRead(SPRINKLER_PIN) == HIGH) ? "ON" : "OFF";
      String message = "Temperature: " + String(temperature) + " °C\n";
      message += "Humidity: " + String(humidity) + " %\n";
      message += "Sprinklers: " + sprinklerState;

      bot.sendMessage(chat_id, message, "");
    } else {
      bot.sendMessage(chat_id, "Unknown command", "");
    }
  }
}



void loop() {
  // Read temperature and humidity
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();

  // Check if the readings are valid
  if (isnan(temperature) || isnan(humidity)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }

  // Get the state of the sprinkler
```

```
  String sprinklerState = (digitalRead(SPRINKLER_PIN) == HIGH) ? "ON" : "OFF";

  // Prepare message
  String message = "Temperature: " + String(temperature) + " °C\n";
  message += "Humidity: " + String(humidity) + " %\n";
  message += "Sprinklers: " + sprinklerState;

  // Send the message to Telegram
  //bot.sendMessage(CHAT_ID, message, "");

  if (millis() > lastTimeBotRan + botRequestDelay)  {
    int numNewMessages = bot.getUpdates(bot.last_message_received + 1);

    while(numNewMessages) {
      Serial.println("got response");
      handleNewMessages(numNewMessages);
      numNewMessages = bot.getUpdates(bot.last_message_received + 1);
    }
    lastTimeBotRan = millis();
  }

  // Wait for a while before sending the next update
  delay(60000); // Update every 60 seconds
}
```

12. Bibliografía

1. Documentación del ESP32:

- Espressif Systems. (n.d.). ESP32 Technical Reference Manual. Espressif Systems. Retrieved from [https://www.espressif.com](https://www.espressif.com/en/products/hardware/esp32/resources)

- Espressif Systems. (n.d.). ESP32 Data Sheet. Espressif Systems. Retrieved from [https://www.espressif.com](https://www.espressif.com/en/products/hardware/esp32/resources)

2. Bibliotecas y Recursos de Programación:

- UniversalTelegramBot Library by Brian Lough. (n.d.). Universal Telegram Bot Library for Arduino. Retrieved from https://github.com/witnessmenow/Universal-Arduino-Telegram-Bot

- ArduinoJson. (n.d.). ArduinoJson - A JSON library for Arduino. Retrieved from https://arduinojson.org

- DHT Sensor Library. (n.d.). DHT Sensor Library by Adafruit. Retrieved from https://github.com/adafruit/DHT-sensor-library

3. WiFi y Seguridad:

- WiFi.h and WiFiClientSecure.h documentation for ESP32: Espressif Systems. (n.d.). ESP-IDF Programming Guide. Retrieved from https://docs.espressif.com/projects/esp-idf/en/latest/esp32/

4. Telegram Bot API:

- Telegram. (n.d.). Bot API Documentation. Telegram. Retrieved from https://core.telegram.org/bots/api

5. DHT22 Sensor:

- Adafruit. (n.d.). DHT22 (AM2302) Sensor Overview. Retrieved from https://learn.adafruit.com/dht/overview

6. Ejemplos y Tutoriales de Código:

- Brian Lough's GitHub Repository. (n.d.). Examples and tutorials for ESP32 and Telegram Bots. Retrieved from https://github.com/witnessmenow

7. Guías y Tutoriales:

- Random Nerd Tutorials. (2020). How to Control a Relay with an ESP32 and Telegram Bot. Retrieved from https://randomnerdtutorials.com/esp32-telegram-bot-relay/

8. Certificados y Seguridad:

- Telegram. (n.d.). Telegram Certificates for Secure Communication. Retrieved from https://core.telegram.org/bots/api#tls

9. NTP Time Retrieval:

- NTP.org. (n.d.). Network Time Protocol (NTP) Documentation. Retrieved from http://www.ntp.org