Sistema de Monitoreo Inteligente para Agricultura Urbana

Proyecto con MQTT, ESP32, Raspberry Pi y Redis

INFORMACIÓN GENERAL DEL PROYECTO

Título: Sistema de Monitoreo y Control Automatizado para Cultivos Urbanos (SMART-GROW)

Nivel: Intermedio-Avanzado

Áreas de conocimiento: IoT, Sistemas Embebidos, Bases de Datos, Comunicaciones, Automatización

---

DESCRIPCIÓN DEL PROYECTO

Objetivo General

Desarrollar un sistema integral de monitoreo y control automatizado para cultivos urbanos utilizando tecnologías IoT, que permita supervisar condiciones ambientales, automatizar el riego y generar reportes de análisis de datos en tiempo real.

Objetivos Específicos

1. Implementar una red de sensores IoT para monitoreo ambiental
2. Desarrollar un sistema de comunicación eficiente usando MQTT
3. Crear una base de datos para alacenamiento y análisis de datos históricos
4. Implementar un sistema de control automatizado de riego
5. Desarrollar una interfaz web para visualización y control
6. Generar reportes y análisis predictivos

ARQUITECTURA DEL SISTEMA

Componentes Principales

1. Nodos Sensores (ESP32)

- Múltiples unidades ESP32 distribuidas en el cultivo
- Sensores integrados por nodo
- Comunicación vía WiFi/MQTT

2. Controlador Central (Raspberry Pi 3)

- Broker MQTT local
- Servidor web y API REST
- Procesamiento de datos y lógica de control
- Interfaz con actuadores

3. Base de Datos (Redis)

- Almacenamiento de datos en tiempo real
- Cache de configuraciones
- Persistencia de datos históricos

4. Interfaz Web

- Dashboard de monitoreo
- Panel de control
- Reportes y análisis

ESPECIFICACIONES TÉCNICAS

Hardware Requerido

Nodos Sensores (3-4 unidades):

- ESP32 DevKit V1
- Sensor de humedad del suelo (YL-69)
- Sensor de temperatura y humedad ambiental (DHT22)
- Sensor de luz (LDR)
- Sensor de pH del suelo (opcional)
- Resistencias y jumpers
- Protoboard
- Caja protectora IP65

Controlador Central:

- Raspberry Pi 4 (4GB RAM)
- Tarjeta microSD 32GB Clase 10
- Fuente de alimentación 5V 3A
- Carcasa con ventilación

Sistema de Actuadores:

- Bomba de agua 12V
- Relé de 5V
- Electroválvulas (2-3 unidades)
- Mangueras y conectores

- Fuente de alimentación 12V

Adicionales:

- Router WiFi
- Cables de red
- Multímetro
- Herramientas básicas de electrónica

Software y Tecnologías

Lenguajes de Programación:

- C++ (Arduino IDE para ESP32)
- Python (Raspberry Pi)
- JavaScript (Frontend)
- HTML/CSS (Interfaz web)

Frameworks y Librerías:

- Arduino IDE
- PlatformIO (opcional)
- Flask/FastAPI (Python web framework)
- Redis-py (Python Redis client)
- Paho MQTT (Cliente MQTT)
- Chart.js (Visualización de datos)
- Bootstrap (Frontend framework)

Herramientas:

- Mosquitto MQTT Broker
- Redis Server
- Git (Control de versiones)
- Postman (Testing API)

DESARROLLO POR ETAPAS

ETAPA 1: Configuración del Entorno

Tareas:

1. **Configuración de Raspberry Pi**
    - Instalacion de Raspbian OS
    - Configuración de SSH y WiFi
    - Instalación de Python, pip, Redis

    - Instalación de Mosquitto MQTT Broker
2. **Configuración de ESP32**
    - Instalación de Arduino IDE
    - Configuración de librerías ESP32
    - Instalación de librerías MQTT (PubSubClient)
    - Pruebas básicas de conectividad
3. **Configuración de Redis**
    - Instalacion y configuración
    - Configuración de persistencia
    - Pruebas de conexión

Entregables:

- Documento de configuración del entorno
- Capturas de pantalla de las configuraciones
- Código básico de "Hello World" para ESP32 y Raspberry Pi

ETAPA 2: Desarrollo de Nodos Sensores

Tareas:

1. **Conexión de Sensores**
    - Esquema de conexiones para cada sensor
    - Calibración de sensores
    - Pruebas individuales de cada sensor
2. **Programación de ESP32**

```
// Estructura básica del código
#include <WiFi.h>
#include <PubSubClient.h>
#include <DHT.h>

// Configuración WiFi y MQTT
const char* ssid = "tu_red";
const char* password = "tu_password";
const char* mqtt_server = "raspberry_pi_ip";

// Funciones principales:
// - setup(): Inicialización
// - loop(): Lectura de sensores y envío MQTT
// - reconnect(): Reconexión automática
// - readSensors(): Lectura de todos los sensores
// - publishData(): Publicación de datos vía MQTT
```

3. **Protocolo de Comunicación MQTT**
    - Definición de topics:
        - smartgrow/node1/temperature
        - smartgrow/node1/humidity

        - smartgrow/node1/soil_moisture
        - smartgrow/node1/light_level
        - smartgrow/node1/status
4. **Gestión de Energía**
    - Implementación de deep sleep
    - Optimización de consumo energético

Entregables:

- Código fuente completo para ESP32
- Esquemas de conexión (Fritzing)
- Documentación de calibración de sensores
- Pruebas de funcionamiento

ETAPA 3: Desarrollo del Controlador Central

Tareas:

1. **Servidor MQTT**
    - Configuración de Mosquitto
    - Definición de políticas de seguridad
    - Configuración de logs
2. **Cliente MQTT Python**

```
# Estructura básica
import paho.mqtt.client as mqtt
import redis
import json
from datetime import datetime

class MQTTHandler:
    def __init__(self):
        self.client = mqtt.Client()
        self.redis_client = redis.Redis(host='localhost', port=6379)

    def on_connect(self, client, userdata, flags, rc):
        # Suscripción a topics
        client.subscribe("smartgrow/+/+")

    def on_message(self, client, userdata, msg):
        # Procesamiento de mensajes
        # Almacenamiento en Redis
        # Lógica de control
```

3. **Integración con Redis**
    - Estructura de datos para almacenamiento
    - Implementación de series temporales
    - Configuración de TTL para datos
4. **Lógica de Control Automatizado**

- Algoritmos de decisión para riego
- Configuración de umbrales
- Sistema de alertas

Entregables:

- Código Python del controlador central
- Documentación de la API
- Configuración de base de datos
- Pruebas de integración

ETAPA 4: Sistema de Actuadores (Semanas 9-10)

Tareas:

1. **Control de Riego**
    - Conexión de relés y electroválvulas
    - Programación de control GPIO
    - Implementación de rutinas de riego
2. **Interfaz MQTT para Control**
    - Topics de control:
        - smartgrow/control/irrigation/zone1
        - smartgrow/control/irrigation/zone2
        - smartgrow/control/system/mode
3. **Seguridad y Protecciones**
    - Límites de tiempo de riego
    - Detección de fallos
    - Modo manual/automático

Entregables:

- Código de control de actuadores
- Esquemas de conexión de potencia
- Documentación de seguridad
- Pruebas de funcionamiento

ETAPA 5: Desarrollo de la Interfaz Web

Tareas:

1. **API REST**

```
from flask import Flask, jsonify, request
from flask_cors import CORS

app = Flask(__name__)
CORS(app)
```

```
@app.route('/api/sensors/current')
def get_current_data():
    # Retorna datos actuales de Redis

@app.route('/api/sensors/history')
def get_historical_data():
    # Retorna datos históricos

@app.route('/api/control/irrigation', methods=['POST'])
def control_irrigation():
    # Control manual de riego
```

2. **Dashboard Web**
    - Visualización en tiempo real
    - Gráficos históricos
    - Panel de control
    - Configuración de alertas
3. **Responsive Design**
    - Optimización para móviles
    - Interfaz intuitiva
    - Notificaciones push

Entregables:

- Código fuente de la aplicación web
- Documentación de la API
- Capturas de pantalla de la interfaz
- Pruebas de usabilidad

ETAPA 6: Análisis y Reportes

Tareas:

1. **Análisis de Datos**
    - Implementación de métricas
    - Análisis de tendencias
    - Correlación entre variables
2. **Reportes Automatizados**
    - Generación de reportes PDF
    - Envío por email
    - Exportación de datos
3. **Algoritmos Predictivos**
    - Predicción de necesidades de riego
    - Detección de anomalías
    - Optimización de recursos

Entregables:

- Módulo de análisis de datos
- Plantillas de reportes
- Documentación de algoritmos
- Ejemplos de análisis

ETAPA 7: Pruebas y Documentación

Tareas:

1. **Pruebas Integrales**
    - Pruebas de sistema completo
    - Pruebas de resistencia
    - Pruebas de conectividad
2. **Documentación Final**
    - Manual de usuario
    - Documentación técnica
    - Guía de instalación
    - Video demostrativo

ESTRUCTURA DE ARCHIVOS DEL PROYECTO

```
smartgrow-project/
├── hardware/
│   ├── esp32/
│   │   ├── main.ino
│   │   ├── sensors.h
│   │   ├── mqtt_handler.h
│   │   └── config.h
│   ├── raspberry/
│   │   ├── gpio_control.py
│   │   └── hardware_config.py
│   └── schematics/
│       ├── sensor_node.fzz
│       └── actuator_connections.fzz
├── software/
│   ├── mqtt_controller/
│   │   ├── main.py
│   │   ├── mqtt_handler.py
│   │   ├── redis_manager.py
│   │   └── control_logic.py
│   ├── web_interface/
│   │   ├── app.py
│   │   ├── static/
│   │   │   ├── css/
│   │   │   ├── js/
│   │   │   └── images/
│   │   └── templates/
│   │       ├── index.html
```

```
│   │       ├── dashboard.html
│   │       └── control.html
│   └── data_analysis/
│       ├── analytics.py
│       ├── reports.py
│       └── ml_models.py
├── config/
│   ├── mosquitto.conf
│   ├── redis.conf
│   └── system_config.json
├── docs/
│   ├── installation_guide.md
│   ├── user_manual.md
│   ├── technical_documentation.md
│   └── api_reference.md
└── tests/
    ├── unit_tests/
    ├── integration_tests/
    └── system_tests/
```

CRITERIOS DE EVALUACIÓN

Aspectos Técnicos (60%)

- **Funcionamiento del Sistema (20%)**
    - Comunicación MQTT estable
    - Lectura precisa de sensores
    - Control efectivo de actuadores
- **Calidad del Código (20%)**
    - Estructura y organización
    - Comentarios y documentación
    - Manejo de errores
- **Integración de Componentes (20%)**
    - Sincronización entre módulos
    - Persistencia de datos
    - Interfaz web funcional

Aspectos de Proyecto (40%)

- **Documentación (15%)**
    - Claridad y completitud
    - Diagramas y esquemas
    - Manual de usuario
- **Presentación (10%)**

    - Demostración en vivo
    - Explicación técnica
    - Defensa de decisiones
- **Innovación y Mejoras (15%)**
    - Características adicionales
    - Optimizaciones implementadas
    - Propuestas de mejora

RECURSOS ADICIONALES

Habilidades Desarrolladas

- Programación de microcontroladores
- Protocolos de comunicación IoT
- Desarrollo web full-stack
- Administración de bases de datos
- Análisis de datos
- Documentación técnica

EXTENSIONES POSIBLES

Mejoras Avanzadas

1. **Machine Learning**
    - Predicción de patrones de crecimiento
    - Optimización automática de parámetros
2. **Conectividad Avanzada**
    - Integración con servicios cloud
    - Notificaciones móviles
3. **Sensores Adicionales**
    - Cámaras para análisis visual
    - Sensores de CO2
    - Medidores de nutrientes
4. **Escalabilidad**
    - Soporte para múltiples cultivos
    - Red mesh de sensores

- Balanceador de carga

Aplicaciones Comerciales

- Agricultura de precisión
- Invernaderos inteligentes
- Jardines verticales urbanos
- Sistemas hidropónicos