# Justificación del Diseño basado en COCOCYS para SMART-GROW

## 1. Análisis

En esta etapa se identificaron las necesidades y requerimientos del sistema de monitoreo para cultivos urbanos. Se analizaron los factores críticos como las variables ambientales a monitorear (humedad, temperatura, luz), la necesidad de automatizar el riego, y la importancia de disponer de datos en tiempo ral para toma de decisiones. Además, se evaluaron tecnologías adecuadas (ESP32, MQTT, Redis, Raspberry Pi) para cumplir los objetivos.

## 2. Metas

Se definieron objetivos claros y medibles: crear una red de sensores IoT eficiente, implementar comunicación MQTT confiable, almacenar y analizar datos históricos, controlar automáticamente el riego, desarrollar una interfaz web funcional y generar reportes predictivos. Estas metas guían todo el proceso para garantizar que el sistema satisfaga las necesidades detectadas.

## 3. Diseño

Se estructuró una arquitectura modular y escalable con componentes bien definidos: nodos sensores con ESP32, un controlador central con Raspberry Pi actuando como broker MQTT y servidor web, base de datos Redis para manejo de datos en tiempo real y almacenamiento histórico, y una interfaz web para visualización y control. El diseño incluye protocolos de comunicación, estructuras de datos y esquemas eléctricos para sensores y actuadores.

## 4. Desarrollo

Se planificó el desarrollo por etapas, desde la configuración del entorno, programación de nodos sensores, desarrollo del controlador cetral, integración de actuadores, hasta la creación de la interfaz web y módulos de análisis. Se implementan los códigos necesarios en C++ (ESP32), Python (controlador y backend), y JavaScript/HTML (frontend), siguiendo buenas prácticas de programación y pruebas unitarias.

## 5. Implementación

Se despliega el sistema físico conectando sensores y actuadores, configurando el broker MQTT, Redis y el servidor web en Raspberry Pi, y realizando pruebas funcionales integrales. Se verifica la comunicación estable, precisión de sensores, control de riego automático, y la usabilidad de la interfaz web, asegurando que el sistema opere en condiciones reales.

## 6. Mejora

Se contempla la mejora continua mediante el análisis de datos recolectados para detectar fallos, optimizar parámetros de riego y consumo energético. Se proponen extensiones avanzadas como incorporacion de algoritmos predictivos, machine learning para patrones de crecimiento, integración con servicios en la nube y soporte para nuevas funcionalidades y escalabilidad del sistema.