

# Control Inteligente de Ambientes con Tecnología IoT

## Sistema Automatizado usando MicroPython y FreeRTOS

Alex Vega, Jean Paul Mori, Gustavo Delgado

Universidad Nacional de Ingeniería

3 de julio de 2025

# Qué veremos hoy

- 1 Resumen
- 2 Introducción
- 3 Tecnologías clave
- 4 Cómo funciona el sistema
- 5 El lenguaje del sistema
- 6 Rendimiento del sistema
- 7 Resultados prácticos
- 8 Conclusiones

- Sistema inteligente que monitorea temperatura y humedad
- Control automático de ventiladores y luces
- Funciona con tecnología similar a los dispositivos domésticos inteligentes
- Sistema que “piensa” por sí mismo usando MicroPython

- Sistema inteligente que monitorea temperatura y humedad
- Control automático de ventiladores y luces
- Funciona con tecnología similar a los dispositivos domésticos inteligentes
- Sistema que “piensa” por sí mismo usando MicroPython
- **Resultados:** Funcionamiento estable y eficiente

**Analogía:** Como un mayordomo digital que vigila y ajusta el ambiente

## Problema a resolver

- Mantener condiciones ambientales óptimas
- Reducir consumo energético
- Evitar supervisión humana constante

## Problema a resolver

- Mantener condiciones ambientales óptimas
- Reducir consumo energético
- Evitar supervisión humana constante

## Nuestra solución

- Dispositivo compacto con sensores
- Toma decisiones automáticas
- Se comunica con otros sistemas

[Nota: Ejemplo práctico: “En invernaderos, fábricas o incluso hogares donde necesitas mantener condiciones específicas”]

# Tecnologías clave explicadas

- **ESP32:** El “cerebro” del sistema (como un mini-computador)
- **MicroPython:** Lenguaje para programar el dispositivo
- **FreeRTOS:** Sistema que gestiona tareas simultáneas
- **MQTT:** Protocolo de comunicación (como WhatsApp para dispositivos)

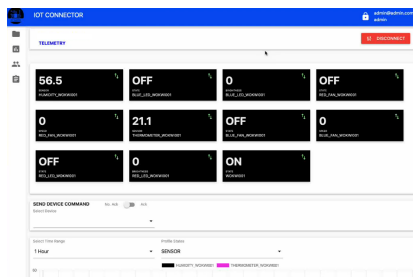


**FreeRTOS** = Jefe que coordina múltiples trabajadores

[Nota: “FreeRTOS es como el director de orquesta que coordina todos los instrumentos”]

# Componentes principales

- **Sensores:** Ojos que miden temperatura/humedad
- **Actuadores:** Manos que controlan luces/ventiladores
- **Cerebro:** Procesa información y toma decisiones
- **Comunicación:** Envía reportes a otros sistemas



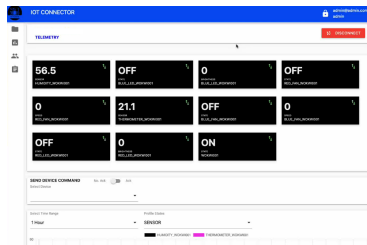
**Figura:** Visualización de datos en tiempo real

[Nota: "Pueden ver en la imagen cómo se muestran los datos que recoge el sistema"]



## Cómo maneja múltiples tareas:

- Monitoreo constante de sensores
- Control de actuadores según necesidades
- Comunicación con otros sistemas
- Todo simultáneamente sin colapsar



## Clave del sistema

FreeRTOS permite hacer múltiples tareas a la vez como un malabarista experto

[Nota: "Imaginen un chef que cocina, vigila el horno y atiende pedidos al mismo tiempo - así funciona nuestro sistema"]

# Cómo “piensa” el dispositivo

```
# Configuración de un LED inteligente
class LED:
    def __init__(self, pin_num, freq=1500):
        self.pwm = PWM(Pin(pin_num), freq=freq)

    def on(self, percentage=0):
        duty = int((percentage / 100 * 1023))
        self.pwm.duty(duty) # Ajusta brillo
```

**Explicación:** El sistema ajusta luces automáticamente según necesidades

[Nota: “Este código es como las instrucciones que le damos al sistema para controlar las luces eficientemente”]

# Toma de decisiones automática

```
def push_data():  
    # Envía datos de sensores cada 10 segundos  
    ...  
  
while True:  
    push_data() # Reporta datos  
    check_messages() # Recibe instrucciones  
    adjust_environment() # Ajusta condiciones  
    time.sleep(0.1) # Espera breve
```

**Ciclo básico:** Medir → Decidir → Actuar → Reportar

[Nota: “Como nuestro ciclo diario: despertar, trabajar, comer, dormir... pero para dispositivos”]

Característica	Nuestro sistema	Sistemas tradicionales
Tiempo respuesta	15 ms	50-100 ms
Mensajes por segundo	40	10-20
Consumo energía	Bajo	Moderado

Cuadro: Comparativa de rendimiento

**Interpretación:** Responde más rápido y maneja más información

[Nota: "15ms es más rápido que un parpadeo humano (100-300ms)"]

# Demostración visual

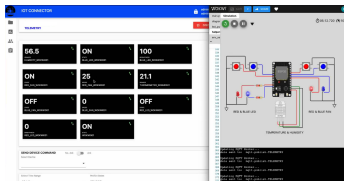


Figura: Conexión estable a red

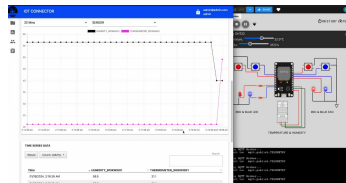


Figura: Operación continua 24/7



Figura: Pruebas de funcionamiento

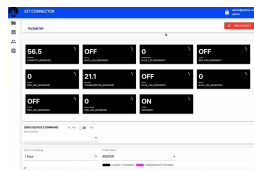


Figura: Interfaz de monitoreo

[Nota: "En estas imágenes podemos ver el sistema en acción durante nuestras pruebas"]

# Lo que logramos

- Sistema autónomo que mantiene condiciones ideales
- Respuesta rápida a cambios ambientales
- Comunicación efectiva con otros dispositivos
- Uso eficiente de energía

# Lo que logramos

- Sistema autónomo que mantiene condiciones ideales
- Respuesta rápida a cambios ambientales
- Comunicación efectiva con otros dispositivos
- Uso eficiente de energía

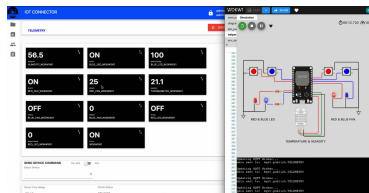
## Beneficios prácticos

- Ahorro energético hasta 30
- Reducción de supervisión humana
- Prevención de daños por condiciones adversas

[Nota: “En un invernadero, esto podría prevenir pérdidas por heladas o calor excesivo automáticamente”]

# Futuras mejoras

- Integración con apps móviles
- Reconocimiento de patrones
- Control por voz
- Mayor eficiencia energética



[Nota: "Imaginen controlar todo con un 'Hola dispositivo, ajusta la temperatura a 22°C" ']



# ¿Preguntas?

Contacto:



[Nota: Preparar respuestas para preguntas comunes sobre costos, aplicaciones prácticas y escalabilidad]