BITACORA PRACTICA No. 3

Dispositivos embebidos con sensores e Interfaces moviles

Nombre: Carlos Alberto Zamudio Velazquez
Matricula: A01799283
Modulo: Integracion de hardware para ciencia de datos
Profesor: Dr. David Higuera Rosales
Periodo: 8 de septiembre - 18 de septiembre, 2025

1. OBJETIVOS DE LA PRACTICA

- Disenar soluciones de dispositivos embebidos con sensores
- Implementar hardware requerido segun especificaciones
- Obtener datos del puerto serie y almacenarlos
- Disenar interfaces basicas de interaccion para moviles
- Integrar Sistema de Monitoreo Ambiental Completo con ML y App Movil

2. MATERIALES UTILIZADOS

- ESP32-C3 DevKit
- Sensor DHT22 (temperatura y humedad)
- LDR (sensor de luz) con resistencia 10kΩ
- LEDs indicadores (azul, verde, rojo) con resistencias 220Ω
- Protoboard y cables jumper
- Resistencia 10kΩ pull-up para DHT22

3. BITACORA DIARIA

3.1. Domingo 8 de Septiembre, 2025

Analisis y Planificacion del Proyecto
Se realizo el analisis inicial de requerimientos y la planificacion del sistema:

- Analisis de requerimientos de la practica
- Investigacion sobre ESP32-C3 y sus capacidades
- Planificacion de arquitectura del sistema completo
- Definicion de tecnologias: ESP32-C3 + FastAPI + Flutter + PostgreSQL

Decisiones tecnicas del equipo:

- Seleccionar ESP32-C3 por capacidades WiFi nativas
- Usar Flutter en lugar de React Native por mejor rendimiento en graficos

- PostgreSQL en lugar de CSV para almacenamiento robusto y consultas complejas
- FastAPI para backend con capacidades ML integradas

3.2. Lunes 9 de Septiembre, 2025

Configuracion de Hardware
Se realizaron las conexiones del hardware segun las especificaciones del ESP32-C3:
Conexiones implementadas:

- DHT22: VCC → 3.3V, DATA → GPIO_3, GND → GND
- Resistencia pull-up 10kΩ entre DATA y VCC del DHT22
- LDR: Un extremo → 3.3V, otro extremo → ADC_CHANNEL_4
- Resistencia 10kΩ desde ADC_CHANNEL_4 a GND (divisor de voltaje)
- LEDs: Azul (GPIO_5), Verde (GPIO_6), Rojo (GPIO_7) con resistencias 220Ω

Pruebas realizadas por el equipo:

- Verificacion de lecturas DHT22: Precision ±0.5°C, ±3 % humedad
- Calibracion LDR: Rango 0-4095 mapeado a condiciones de luz
- Pruebas de conectividad WiFi exitosas

Formulas utilizadas para calibracion:

- Voltaje LDR: $V_{LDR} = \frac{ADC_{value} \times 3{,}3V}{4095}$
- Resistencia LDR: $R_{LDR} = \frac{V_{LDR} \times 10k\Omega}{3{,}3V - V_{LDR}}$

3.3. Martes 10 de Septiembre, 2025

Desarrollo del Firmware ESP32-C3
Se implemento el sistema embebido utilizando ESP-IDF framework:
Caracteristicas implementadas:

- Configuracion de tareas FreeRTOS para lectura no bloqueante de sensores
- Protocolo HTTP POST con autenticacion mediante header ”protected”
- Sistema de reconexion WiFi automatica con event handlers
- Logica de indicadores LED basada en umbrales de temperatura

Mi contribucion principal en el microcontrolador:

- Implemente la logica de envio de datos JSON al servidor FastAPI
- Configure el sistema de LEDs indicadores con umbrales personalizables
- Ajuste los parametros de muestreo de sensores y timing
- Optimice el formato de datos para compatibilidad con el backend

Estructura de datos implementada:

```
{
  "source": "ESP32",
  "sensor": "temperature|humidity|light",
  "value": <float_value>
}
```

Logica de LEDs programada:

- LED Azul: Temperatura <20°C (frio)
- LED Verde: Temperatura entre 20°C y 30°C (normal)
- LED Rojo: Temperatura >30°C (caliente)

3.4. Miercoles 11 de Septiembre, 2025

Desarrollo del Backend FastAPI
El equipo implemento el servidor backend con las siguientes caracteristicas:
Arquitectura del servidor:

- Middleware de autenticacion con header ”protected”personalizado
- Base de datos PostgreSQL con indices optimizados para time-series
- Modelos Pydantic para validacion de datos de entrada y respuesta
- Sistema de logging estructurado con diferentes niveles

Esquema de base de datos implementado:

```
CREATE TABLE metrics (
    id SERIAL PRIMARY KEY,
    timestamp TIMESTAMPTZ DEFAULT NOW(),
    source VARCHAR(100),
    sensor VARCHAR(50),
    value FLOAT
);

-- Indices para optimizar consultas temporales
CREATE INDEX idx_sensor_timestamp ON metrics(sensor, timestamp);
CREATE INDEX idx_timestamp_sensor ON metrics(timestamp, sensor);
```

Endpoints de API implementados:

- POST /metric - Recepcion de datos de sensores desde ESP32
- GET /metrics/history - Consulta historica con filtros temporales
- WebSocket /ws-metrics - Stream de datos en tiempo real
- GET /predict/{sensor_type} - Predicciones ML multi-horizonte

3.5. Jueves 12 de Septiembre, 2025

Implementacion Machine Learning
Se desarrollo el sistema de predicciones usando LightGBM:
Sistema de predicciones implementado:

- Entrenamiento automatico cada 6 horas con minimo 10 puntos de datos
- Caracteristicas temporales: medias moviles, tendencias, estacionalidad
- Predicciones multi-horizonte: 15min, 1h, 6h, 24h
- Calculo de intervalos de confianza basados en varianza historica

Feature Engineering aplicado:

```
def create_features(data):
    features = [
        data['value'].rolling(5).mean().iloc[-1],    # Media 5 puntos
        data['value'].rolling(10).mean().iloc[-1],   # Media 10 puntos
        (data['value'].iloc[-1] - data['value'].iloc[-5]) / 5,  # Tendencia
        datetime.now().hour,                         # Hora del dia
        datetime.now().weekday()                     # Dia de la semana
    ]
    return np.array(features)
```

Metricas de rendimiento obtenidas:

- MAE temperatura: 1.1°C para predicciones 1h
- MAE humedad: 4.2 % para predicciones 1h
- Tiempo de inferencia: <50ms promedio
- Precision 15min: 96 % dentro de ±1°C

3.6. Viernes 13 de Septiembre, 2025

Desarrollo Aplicacion Flutter
Se desarrollo la aplicacion movil con arquitectura modular:
Arquitectura implementada:

- Patron MVC con servicios separados para API y WebSocket
- Gestion de estado reactiva con StreamBuilder para datos en tiempo real
- Biblioteca fl_chart para visualizaciones interactivas
- Sistema de notificaciones locales con permission_handler

Servicios implementados en la app:

- ApiService - Cliente HTTP con autenticacion automatica
- WebSocketService - Conexion persistente con reconexion automatica
- LocalNotificationService - Alertas push locales

- LocalAlarmService - Sistema de alarmas configurables

Mi contribucion principal en la aplicacion:

- Implemente el sistema de graficos interactivos con fl_chart
- Configure las notificaciones push y alarmas locales
- Desarrolle la logica de reconexion automatica del WebSocket
- Diseñe la interfaz de usuario del dashboard principal
- Integre el sistema de predicciones ML en la UI

Pantallas principales desarrolladas:

- Dashboard con cards de sensores actualizadas en tiempo real
- Graficos de linea temporal con zoom y pan interactivo
- Pantalla de predicciones ML con intervalos de confianza
- Configuracion de alarmas personalizables por sensor

3.7. Sabado 14 de Septiembre, 2025

Integracion WebSocket Tiempo Real
Se implemento el sistema de comunicacion bidireccional:
Flujo de datos en tiempo real:

1. ESP32-C3 envia datos via HTTP POST cada 5 segundos
2. FastAPI almacena en PostgreSQL y encola para distribución WebSocket
3. Flutter recibe actualizaciones y actualiza UI reactivamente
4. Sistema evalua umbrales y dispara notificaciones automaticamente

Optimizaciones implementadas:

- Cola asincrona en FastAPI para distribucion eficiente
- Reconexion automatica en Flutter con backoff exponencial
- Buffer local para manejo de desconexiones temporales
- Compresion de datos para optimizar ancho de banda

3.8. Domingo 15 de Septiembre, 2025

Sistema de Alertas y Notificaciones
Se completo el sistema de alertas multi-modal:
Alertas hardware (ESP32-C3):

- Logica de LEDs implementada segun umbrales de temperatura
- Frecuencia de parpadeo proporcional a desviacion del rango normal
- Estados visuales: Frio (azul), Normal (verde), Caliente (rojo)

Notificaciones moviles implementadas:

- Push notifications para valores criticos de sensores
- Alarmas programables con intervalos personalizables
- Persistencia en SharedPreferences para configuracion de usuario
- Multiples umbrales configurables por tipo de sensor

3.9. Lunes 16 de Septiembre, 2025

Pruebas y Optimizacion del Sistema
Se realizaron pruebas exhaustivas del sistema completo:
Pruebas de sistema ejecutadas:

- Conectividad: 24h continuas sin desconexiones criticas
- Carga: 50 clientes WebSocket simultaneos sin degradacion
- Precision ML: Validacion cruzada con 500+ puntos historicos
- Autonomia: Aplicacion movil funcional durante 6h+ de uso continuo

Optimizaciones aplicadas:

- Indices compuestos en BD para consultas temporales (90 % mejora)
- Lazy loading en graficos historicos de Flutter
- Pool de conexiones PostgreSQL para alta concurrencia

Resultados de rendimiento obtenidos:

- Latencia API: 45ms promedio para endpoints basicos
- Throughput: 800+ requests/segundo en condiciones normales
- Tiempo real completo: 180ms desde sensor hasta aplicacion
- Disponibilidad del sistema: 99.8 % durante pruebas de 48h

3.10. Martes 17 de Septiembre, 2025

Documentacion
Validacion de requerimientos completada:

Datos ambientales almacenados (PostgreSQL supera CSV)

API REST funcional con capacidades ML avanzadas

Aplicacion movil Flutter completamente funcional

Sistema de alertas multi-modal (LED + notificaciones)

Analisis predictivo con metricas de confianza

4. ARQUITECTURA FINAL DEL SISTEMA

ESP32-C3	FastAPI Server	Flutter App
DHT22 + LDR	PostgreSQL	Dashboard
LEDs indicadores	Machine Learning	Graficos tiempo real
WiFi + HTTP	WebSocket	Notificaciones

Flujo de datos implementado:
Sensores → ESP32-C3 → HTTP POST → FastAPI → PostgreSQL
FastAPI → WebSocket → Flutter App → Usuario

5. PROBLEMAS ENCONTRADOS Y SOLUCIONES

5.1. Desafios de Hardware

Problema: Lecturas erraticas iniciales del DHT22
Solucion aplicada: Agregar resistencia pull-up 10kΩ y delay de inicializacion de 2 segundos
Problema: Ruido en lecturas ADC del LDR
Solucion aplicada: Promedio de 10 lecturas consecutivas con filtro de mediana
Problema: Desconexiones WiFi esporadicas
Solucion aplicada: Event handlers para reconexion automatica con retry exponencial

5.2. Desafios de Software

Problema: Desconexiones WebSocket frecuentes en la aplicacion
Solucion aplicada: Implementar reconexion automatica con backoff exponencial
Problema: Latencia alta en consultas historicas de gran volumen
Solucion aplicada: Indices compuestos en BD y paginacion de resultados
Problema: Lag en graficos de Flutter con muchos puntos
Solucion aplicada: Lazy loading y decimacion inteligente de datos

6. CONCLUSIONES

6.1. Objetivos Alcanzados

- Sistema completo funcional integrando hardware, backend y aplicacion movil
- Reemplazo exitoso de CSV con base de datos PostgreSQL
- Implementacion de Machine Learning predictivo
- Desarrollo de aplicacion Flutter
- Sistema de alertas multi-modal (LED + notificaciones) funcional
- Comunicacion tiempo real WebSocket estable y confiable

6.2. Aprendizajes Tecnicos Principales

- Dominio de ESP32-C3 y sensores ambientales de precision
- Desarrollo de APIs REST asincronas con FastAPI

- Implementacion de modelos ML para prediccion de series temporales
- Desarrollo de aplicaciones moviles nativas con Flutter
- Integracion completa de sistemas IoT complejos
- Optimizacion de bases de datos para aplicaciones time-series

6.3. Contribuciones Personales Destacadas

En el microcontrolador ESP32-C3:

- Logica de envio de datos JSON estructurado
- Sistema de LEDs indicadores con umbrales configurables
- Optimizacion de parametros de muestreo

En la aplicacion Flutter:

- Sistema de graficos interactivos con fl_chart
- Logica de notificaciones push y alarmas
- Reconexion automatica del WebSocket
- Diseño de interfaz de usuario del dashboard
- Integracion de predicciones ML en la UI

Este proyecto demuestra la integracion exitosa de tecnologias embebidas, backend moderno y aplicaciones moviles para crear un sistema de monitoreo ambiental completo y escalable.