



TECNICATURA SUPERIOR EN  
Telecomunicaciones



## ARQUITECTURA Y CONECTIVIDAD

### Módulo II: Arquitectura en Redes IoT Comunicaciones de Bajo Consumo

**Bienvenido a las prácticas de Arquitectura y Conectividad:**

**La modalidad será la siguiente:**

Cada práctica se desarrollará en forma grupal, debiendo subir e desarrollo de la misma al repositorio (respetando la estructura de monorepositorio) establecido por grupo. Los ejercicios serán implementados de forma que a cada integrante le corresponda 1 o más tareas (issues); por lo que deberán crear el proyecto correspondiente con la documentación asociada si hiciera falta, y asignar los issues por integrante. De esta forma quedará documentada la colaboración de cada alumno.

#### Actividad:

- 1) Implementen una simulación o un prototipo físico de una Conexión en RF (radio frecuencia) mediante LoRaWan en Wokwi o Proteus, utilizando **ESP32** ó **ARDUINO** con las siguientes especificaciones:
  - a. Un ESP32 ó ARDUINO en modo Transmisor (TX).
  - b. Un ESP32 ó ARDUINO en modo Receptor (RX).
  - c. Un sensor de Temperatura y Humedad
  - d. Uno o dos Relay's para el comando de una lámpara de potencia
- 2) Colocar en el repositorio todos los catasheet's correspondientes.
- 3) Realizar una presentación en \*.pptx, Canvas o software de su elección, con los pasos que siguieron para llegar al resultado final. La presentación no debe tener más de 10 diapositivas.

# ARQUITECTURA Y CONECTIVIDAD

## Simulación con LoRa



- **Nombre del proyecto:** “*Simulación de red LoRa con ESP32*”.
- **Objetivo:** Comunicación inalámbrica de sensores de bajo consumo.

### Datasheets de Componentes

1. **ESP32 DevKit V1**
  - ESP32 Datasheet (Espressif)
2. **Módulo LoRa SX1278 (RA-02)**
  - SX1278 Datasheet (Semtech)
3. **Sensor de Temperatura y Humedad**
  - DHT11 Datasheet
  - DHT22 Datasheet
4. **Módulo de Relé 5V**
  - 5V Relay Module Datasheet
5. **Display OLED SSD1306**
  - SSD1306 Datasheet

## Configuración del Proyecto en PlatformIO

Para organizar tu proyecto en PlatformIO, puedes estructurarlo de la siguiente manera:

CSS

```

LoRaProject/
├── tx/                                # Código del transmisor
│   ├── src/
│   │   ├── main.cpp
│   │   └── platformio.ini
│   └── rx/                            # Código del receptor
│       ├── src/
│       │   ├── main.cpp
│       │   └── platformio.ini
└──
  
```

## Archivo platformio.ini (tanto para TX como RX):

```

ini
[env:esp32dev]
platform = espressif32
board = esp32dev
framework = arduino
lib_deps =
    sandeepmistry/LoRa@^0.8.0
    adafruit/DHT sensor library@^1.4.4
    adafruit/Adafruit SSD1306@^2.4.3
    adafruit/Adafruit GFX Library@^1.10.10
  
```

## Código para el Transmisor (TX)

```

cpp
#include <SPI.h>
#include <LoRa.h>
#include <DHT.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#define DHTPIN 4
#define DHTTYPE DHT22
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET -1
DHT dht(DHTPIN, DHTTYPE);
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
void setup() {
    serial.begin(115200);
    dht.begin();
    if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
        Serial.println(F("Fallo al iniciar el display OLED"));
        while(true);
    }
    display.clearDisplay();
    display.setTextSize(1);
    display.setTextColor(SSD1306_WHITE);
    LoRa.setPins(5, 14, 2); // NSS, RST, DIO0
    if (!LoRa.begin(915E6)) {
        Serial.println("Error iniciando LoRa");
        while (1);
    }

    Serial.println("Transmisor iniciado");
}

void loop() {
  
```

```
float t = dht.readTemperature();
float h = dht.readHumidity();

if (isnan(t) || isnan(h)) {
    Serial.println("Error leyendo el sensor DHT!");
    delay(2000);
    return;
}

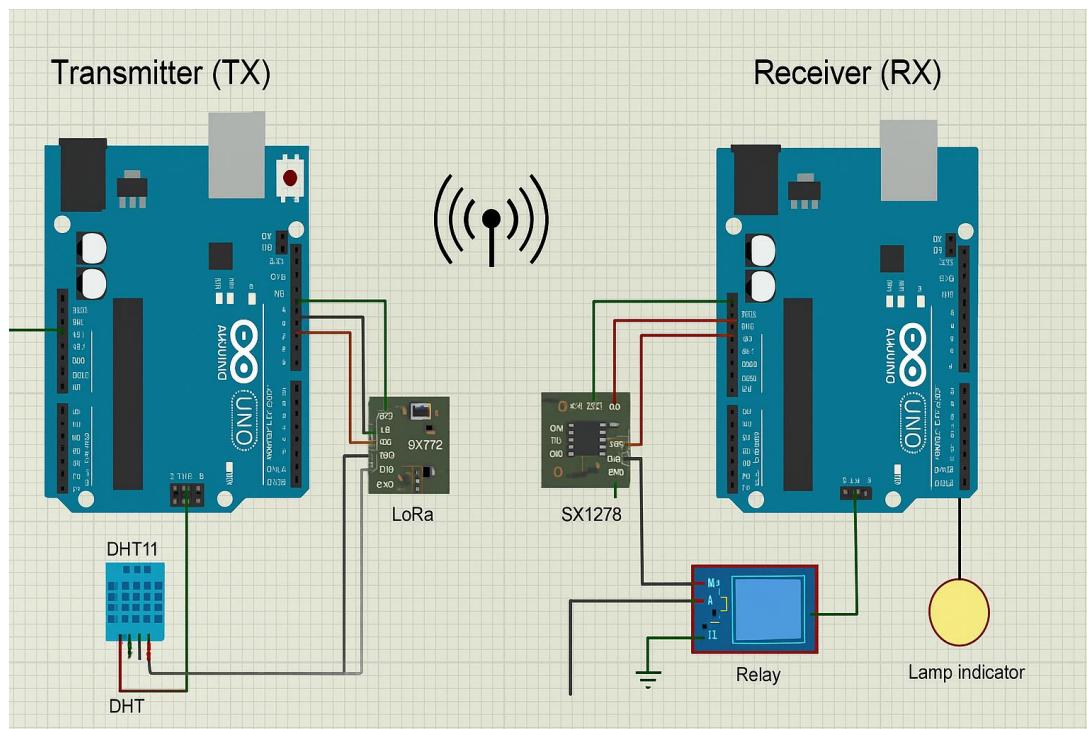
String mensaje = "Temp:" + String(t) + "C Hum:" + String(h) + "%";
Serial.println("Enviando -> " + mensaje);
LoRa.beginPacket();
LoRa.print(mensaje);
LoRa.endPacket();
display.clearDisplay();
display.setCursor(0,0);
display.println("Transmisor");
display.println(mensaje);
display.display();
delay(5000);
}
```

## Código para el Receptor (RX)

```
cpp
#include <SPI.h>
#include <LoRa.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#define RELAY_PIN 12
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET -1
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
void setup() {
    Serial.begin(115200);
    pinMode(RELAY_PIN, OUTPUT);
    digitalWrite(RELAY_PIN, LOW); // Apagado por defecto
    if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
        Serial.println(F("Fallo al iniciar el display OLED"));
        while(true);
    }
    display.clearDisplay();
    display.setTextSize(1);
    display.setTextColor(SSD1306_WHITE);
    LoRa.setPins(5, 14, 2);
    if (!LoRa.begin(915E6)) {
        Serial.println("Error iniciando LoRa");
        while (1);
    }
    Serial.println("Receptor iniciado");
}
void loop() {
    int packetSize = LoRa.parsePacket();
    if (packetSize) {
        String recibido = "";
        while (LoRa.available()) {
            recibido += (char)LoRa.read();
        }
    }
}
```

```
Serial.println("Recibido: " + recibido);
display.clearDisplay();
display.setCursor(0,0);
display.println("Receptor");
display.println(recibido);
display.display();
int tempIndex = recibido.indexOf("Temp:");
int humIndex = recibido.indexOf("C");
if (tempIndex >= 0 && humIndex > tempIndex) {
    float temp = recibido.substring(tempIndex + 5, humIndex).toFloat();
    if (temp > 30) {
        digitalWrite(RELAY_PIN, HIGH);
    } else {
        digitalWrite(RELAY_PIN, LOW);
    }
}
}
```

## Configuración del circuito .DSN con Proteus



- **Microcontrolador:** Arduino UNO
- **Sensor DHT22:**
  - Data conectado al pin **D2** de Arduino
- **Display LCD 16x2** (reemplazo del OLED SSD1306):
  - Conectado mediante **modo 4 bits**
  - Pines RS, E, D4–D7 conectados a D7, D6, D5, D4, D3 respectivamente
- **Alimentación:** VCC y GND correctamente conectados
- **Pull-up resistor de 10kΩ** en la línea de datos del DHT22

El LCD mostrará:



makefile

Temp: XX.X C

Humedad: XX.X %

- El archivo .DSN de Proteus con esta configuración.

### Pruebas con Monitor Serial

Asegurar de tener abierto el **Monitor Serial** en ambos dispositivos (TX y RX). Se puede Ver el mensaje enviado desde TX.

- Confirmar que RX lo recibe.
- Ver si el relé cambia de estado (HIGH/LOW).
- Visualizar los datos en el display OLED.

### 1.Nombre del proyecto: “*Simulación de red LoRa con ESP32*”.

Objetivo: Comunicación inalámbrica de sensores de bajo consumo.

### 2. Componentes Utilizados

Mostrar una diapositiva o toma con:

- ESP32
- Módulo LoRa SX1278
- Sensor DHT22
- Módulo de relé
- Display OLED

### 3. Esquema de Conexión

- Mostrar gráfico tipo Proteus o dibujo de conexiones.
- Explicar brevemente:
  - “TX: Sensor mide temperatura/humedad → LoRa transmite.”
  - “RX: Recibe → muestra en display y activa relé si  $T > 30^{\circ}\text{C}$ .”

### 4. Simulación o Demostración en Vivo

- Mostrar:
  - Código cargado desde PlatformIO.
  - Datos en el Monitor Serial.
  - Display OLED con los datos.
  - Encendido del relé al superar el umbral.

### 5. Organización del Código

- TX y RX separados.
- Librerías usadas.

#### DHT.h

Para leer temperatura y humedad del sensor DHT11 o DHT22.

#### Adafruit\_Sensor.h

Librería base para sensores compatibles.

#### Adafruit\_SSD1306.h y Adafruit\_GFX.h

Para controlar la pantalla OLED I2C y mostrar gráficos o texto.

#### LoRa.h

Para manejar la comunicación entre módulos LoRa SX1278 mediante SPI.  
(En simulación Proteus)

Se puede usar LiquidCrystal.h como alternativa visual si OLED no está disponible.

Lógica: lectura, envío, recepción, decisión.

## 6. Conclusiones

### **Comunicación Inalámbrica Eficiente**

La tecnología LoRa permite establecer una comunicación inalámbrica de largo alcance entre dispositivos IoT con bajo consumo energético, ideal para entornos donde no se dispone de WiFi constante.

### **Monitoreo Remoto en Tiempo Real**

El sistema transmite datos de temperatura y humedad desde un sensor hacia otro módulo receptor, facilitando el monitoreo remoto y confiable de variables ambientales.

### **Visualización de Datos en Pantallas OLED**

Ambas unidades (transmisor y receptor) cuentan con una interfaz visual clara y compacta que mejora la experiencia del usuario al mostrar información en tiempo real.

### **Automatización Mediante Relé**

El uso de un relé controlado por la temperatura permite integrar el sistema con dispositivos eléctricos, lo que abre la puerta a funciones automáticas como encendido de ventiladores, extractores, calefactores, etc.

### **Versatilidad de ESP32 (o Arduino)**

Aunque el proyecto está diseñado con ESP32, su simulación con Arduino en Proteus demuestra la flexibilidad del enfoque y permite validar la lógica antes de la implementación final.

### **Aprendizajes**

**Manejo de sensores digitales (DHT11/DHT22)** para capturar datos ambientales.

**Programación de módulos de visualización (OLED/I2C)** para mostrar datos de forma eficiente.

**Configuración y simulación de redes LoRa (teórica en Proteus)** para comprender su funcionamiento.

**Control de actuadores (relé) según condiciones programadas**, fundamental en domótica e IoT.

**Importancia de la arquitectura modular:** separar funciones entre transmisión, visualización y control hace que el sistema sea más robusto y escalable.

**BIBLIOGRAFIA**

- Adafruit Industries. (n.d.). Adafruit SSD1306 OLED display library. GitHub.  
[https://github.com/adafruit/Adafruit\\_SSD1306](https://github.com/adafruit/Adafruit_SSD1306)
- Adafruit Industries. (n.d.). Adafruit GFX library. GitHub.  
<https://github.com/adafruit/Adafruit-GFX-Library>
- Mistry, S. (n.d.). LoRa library for Arduino. GitHub.  
<https://github.com/sandeepmistry/arduino-LoRa>
- Adafruit Industries. (n.d.). DHT sensor library for Arduino. GitHub.  
<https://github.com/adafruit/DHT-sensor-library>
- Proteus Design Suite. (n.d.). Proteus PCB Design and Simulation Software.  
Labcenter Electronics. <https://www.labcenter.com/>
- Espressif Systems. (n.d.). ESP32 Series Datasheet.  
<https://www.espressif.com/en/products/socs/esp32/resources>
- Arduino. (n.d.). Arduino IDE. <https://www.arduino.cc/en/software>