**Módulos LoRa**

**Modelos de Módulos LoRa  
  
Semtech SX1276/77/78/79**

* **Ficha Técnica:** [**Semtech SX1276**](datasheet/sx1276_77_78_79.pdf)
* **Aplicación:** Utilizado para comunicación LoRa en proyectos de largo alcance. Ideal para redes de sensores IoT en áreas extensas.
* **Ventajas:**
  + **Alcance Extendido:** Hasta 15-20 km en áreas rurales**.**
  + **Bajo Consumo Energético:** Adecuado para dispositivos con batería.
* **Desventajas:**
  + **Ancho de Banda Limitado:** No adecuado para alta velocidad de datos.
  + **Latencia Alta:** Puede ser más alta que otras tecnologías.

**Dragino LoRa Shield**

* **Ficha Técnica:** [**Dragino LoRa Shield**](datasheet/Datasheet_LoraShield.pdf)
* **Aplicación**: Compatible con Arduino y otros microcontroladores. Ideal para prototipos y proyectos de pequeña escala.
* **Ventajas:**
  + **Compatibilidad**: Fácil de usar con Arduino.
  + **Costo:** Relativamente bajo.
* **Desventajas:**
  + **Alcance Limitado en entornos urbanos:** Puede ser menor comparado con otros módulos de LoRa.

**HopeRF RFM95W**

* **Ficha Técnica:** [**HopeRF RFM95W**](datasheet/RFM300HW%20RFM300W Datasheet EN%20V1.31.pdf)
* **Aplicación:** Módulo LoRa con un alcance y sensibilidad mejorados, adecuado para proyectos de largo alcance.
* **Ventajas:**
  + **Sensibilidad Alta:** Mejora el alcance y la calidad de la señal.
  + **Bajo Consumo:** Ideal para aplicaciones con batería.
* **Desventajas:**
  + **Requiere Circuitos Adicionales:** Puede necesitar circuitos externos para manejo de energía y conectividad.

**Murata CMWX1ZZABZ**

* **Ficha Técnica**: Murata CMWX1ZZABZ
* **Aplicación**: Módulo LoRa con un diseño compacto y bajo consumo, adecuado para dispositivos portátiles y aplicaciones IoT.
* **Ventajas**:
  + **Compacto**: Tamaño reducido, ideal para proyectos con espacio limitado.
  + **Bajo Consumo Energético**: Eficiente en términos de consumo de energía.
* **Desventajas**:
  + **Costo**: Puede ser más caro en comparación con otros módulos.
  + **Interfaz de Programación**: Requiere atención especial en la programación y configuración.

**Adafruit RFM95W**

* **Ficha Técnica**: [Adafruit RFM95W](https://learn.adafruit.com/adafruit-rfm69hcw-and-rfm96-rfm95-rfm98-lora-packet-padio-breakouts/downloads)
* **Aplicación**: Módulo LoRa diseñado para ser usado con plataformas de prototipado como Arduino y Raspberry Pi.
* **Ventajas**:
  + **Facilidad de Uso**: Documentación y soporte extensos.
  + **Alcance y Sensibilidad**: Buena sensibilidad y alcance.
* **Desventajas**:
  + **Tamaño**: Más grande comparado con módulos más compactos.
  + **Costo**: Más caro en comparación con módulos más básicos.

**SparkFun LoRa Gateway**

* **Ficha Técnica**: SparkFun LoRa Gateway
* **Aplicación**: Diseñado para funcionar como una puerta de enlace LoRa, adecuado para proyectos que requieren un gateway central.
* **Ventajas**:
  + **Gateway Completo**: Incluye funcionalidad completa para actuar como un gateway.
  + **Facilidad de Integración**: Buen soporte y documentación.
* **Desventajas**:
  + **Costo**: Más caro debido a la funcionalidad adicional.
  + **Complejidad**: Mayor complejidad en comparación con módulos simples.

**Libelium Waspmote LoRa**

* **Ficha Técnica**: Libelium Waspmote LoRa
* **Aplicación**: Módulo LoRa con integración en la plataforma Waspmote para aplicaciones industriales y de infraestructura.
* **Ventajas**:
  + **Robustez**: Diseñado para aplicaciones industriales.
  + **Integración**: Bien integrado con la plataforma Waspmote.
* **Desventajas**:
  + **Costo**: Alto costo.
  + **Interfaz Compleja**: Requiere familiaridad con la plataforma Waspmote.

**2. Conexión y Programación con ESP32**

**Conexión del Semtech SX1276 al ESP32**

1. **Conexiones Físicas:**
   * **VCC del SX1276 a 3.3V del ESP32.**
   * **GND del SX1276 a GND del ESP32.**
   * **SCK del SX1276 a GPIO 18 (SCK) del ESP32.**
   * **MISO del SX1276 a GPIO 19 (MISO) del ESP32.**
   * **MOSI del SX1276 a GPIO 23 (MOSI) del ESP32.**
   * **NSS del SX1276 a GPIO 5 (CS) del ESP32.**
   * **DIO0 del SX1276 a GPIO 4 del ESP32.**
2. **Programación (Uso de la librería RadioHead):**

**cpp**

**#include <SPI.h>**

**#include <RH\_RF95.h>**

**#define RF95\_FREQ 915.0**

**#define RFM95\_CS 5**

**#define RFM95\_RST 14**

**#define RFM95\_INT 4**

**RH\_RF95 rf95(RFM95\_CS, RFM95\_INT);**

**void setup() {**

**Serial.begin(9600);**

**while (!Serial) { }**

**if (!rf95.init()) {**

**Serial.println("LoRa init failed");**

**while (1);**

**}**

**rf95.setFrequency(RF95\_FREQ);**

**rf95.setTxPower(23, false);**

**}**

**void loop() {**

**Serial.println("Sending to LoRa");**

**rf95.send((uint8\_t \*)"Hello", 5);**

**rf95.waitPacketSent();**

**delay(1000);**

**}**

**Conexión de Módulo Murata CMWX1ZZABZ al ESP32**

1. **Conexiones Físicas**:
   * **VCC** del Murata CMWX1ZZABZ a **3.3V** del ESP32.
   * **GND** del Murata CMWX1ZZABZ a **GND** del ESP32.
   * **SCK** del Murata CMWX1ZZABZ a **GPIO 18** (SCK) del ESP32.
   * **MISO** del Murata CMWX1ZZABZ a **GPIO 19** (MISO) del ESP32.
   * **MOSI** del Murata CMWX1ZZABZ a **GPIO 23** (MOSI) del ESP32.
   * **NSS** del Murata CMWX1ZZABZ a **GPIO 5** (CS) del ESP32.
   * **DIO0** del Murata CMWX1ZZABZ a **GPIO 4** del ESP32.

**Programación**:

#include <SPI.h>  
#include <LoRa.h>

void setup() {  
Serial.begin(115200);  
while (!Serial);  
 if (!LoRa.begin(915E6)) {  
Serial.println("LoRa init failed");  
while (1);}  
 LoRa.setTxPower(20);}

void loop() {

Serial.println("Sending LoRa message...");

LoRa.beginPacket();

LoRa.print("Hello World");  
 LoRa.endPacket();  
delay(5000);}

**Conexión de Módulo Adafruit RFM95W al ESP32**

1. **Conexiones Físicas**:
   * **VCC** del Adafruit RFM95W a **3.3V** del ESP32.
   * **GND** del Adafruit RFM95W a **GND** del ESP32.
   * **SCK** del Adafruit RFM95W a **GPIO 18** (SCK) del ESP32.
   * **MISO** del Adafruit RFM95W a **GPIO 19** (MISO) del ESP32.
   * **MOSI** del Adafruit RFM95W a **GPIO 23** (MOSI) del ESP32.
   * **NSS** del Adafruit RFM95W a **GPIO 5** (CS) del ESP32.
   * **DIO0** del Adafruit RFM95W a **GPIO 4** del ESP32.

**Programación**:

#include <SPI.h>  
#include <LoRa.h>

void setup() {

Serial.begin(115200);

while (!Serial);

if (!LoRa.begin(915E6)) {

Serial.println("LoRa init failed");

while (1)}  
LoRa.setTxPower(20);}

void loop() {

Serial.println("Sending LoRa message...");  
LoRa.beginPacket();  
LoRa.print("Hello World");  
LoRa.endPacket();  
delay(5000);

}

**Conexión de Módulo SparkFun LoRa Gateway al ESP32**

1. **Conexiones Físicas**:
   * **VCC** del SparkFun LoRa Gateway a **5V** del ESP32.
   * **GND** del SparkFun LoRa Gateway a **GND** del ESP32.
   * **SCK** del SparkFun LoRa Gateway a **GPIO 18** (SCK) del ESP32.
   * **MISO** del SparkFun LoRa Gateway a **GPIO 19** (MISO) del ESP32.
   * **MOSI** del SparkFun LoRa Gateway a **GPIO 23** (MOSI) del ESP32.
   * **NSS** del SparkFun LoRa Gateway a **GPIO 5** (CS) del ESP32.
   * **DIO0** del SparkFun LoRa Gateway a **GPIO 4** del ESP32.

**Programación**:

#include <SPI.h>  
#include <LoRa.h>  
void setup() {

Serial.begin(115200);

while (!Serial);  
 if (!LoRa.begin(915E6)) {

Serial.println("LoRa init failed");

while (1); }

LoRa.setTxPower(20);}

void loop() {

Serial.println("Sending LoRa message...");

LoRa.beginPacket();

LoRa.print("Hello World");

LoRa.endPacket();

delay(5000);

}

**Conexión de Módulo Libelium Waspmote LoRa al ESP32**

1. **Conexiones Físicas**:
   * **VCC** del Libelium Waspmote LoRa a **3.3V** del ESP32.
   * **GND** del Libelium Waspmote LoRa a **GND** del ESP32.
   * **SCK** del Libelium Waspmote LoRa a **GPIO 18** (SCK) del ESP32.
   * **MISO** del Libelium Waspmote LoRa a **GPIO 19** (MISO) del ESP32.
   * **MOSI** del Libelium Waspmote LoRa a **GPIO 23** (MOSI) del ESP32.
   * **NSS** del Libelium Waspmote LoRa a **GPIO 5** (CS) del ESP32.
   * **DIO0** del Libelium Waspmote LoRa a **GPIO 4** del ESP32.

**Programación**:

#include <SPI.h>  
#include <LoRa.h>

void setup() {

Serial.begin(115200);

while (!Serial);

if (!LoRa.begin(915E6)) {

Serial.println("LoRa init failed");

while (1);

}

LoRa.setTxPower(20);

}

void loop() {

Serial.println("Sending LoRa message...");

LoRa.beginPacket();

LoRa.print("Hello World");

LoRa.endPacket();

delay(5000);

}

**Módulos LoRa y Conexión con ESP32**

* **Semtech SX1276**: Excelente para comunicación de largo alcance con bajo consumo energético. Conexión y programación simple.
* **Dragino LoRa Shield**: Ideal para prototipos con Arduino. Facilidad de uso.
* **HopeRF RFM95W**: Buena sensibilidad y alcance, requiere circuitos adicionales.
* **Murata CMWX1ZZABZ**: Compacto y de bajo consumo, pero más caro.
* **Adafruit RFM95W**: Fácil de usar, pero de mayor tamaño y costo.
* **SparkFun LoRa Gateway**: Módulo de gateway completo, más caro y complejo.
* **Libelium Waspmote LoRa**: Robusto para aplicaciones industriales, pero costoso y requiere familiaridad con Waspmote.

**Protocolos MQTT**

* **Descripción**: MQTT es un protocolo de mensajería ligero para sensores y dispositivos móviles optimizado para conexiones de red con ancho de banda limitado.
* **Aplicación**: Utilizado para la comunicación entre dispositivos en redes IoT. Ideal para transmitir datos de sensores y controlar dispositivos a través de Internet.
* **Ventajas**:
  + **Ligero y Eficiente**: Bajo uso de ancho de banda y consumo energético.
  + **Escalabilidad**: Compatible con una gran cantidad de dispositivos.
  + **Calidad del Servicio (QoS)**: Ofrece niveles de QoS para garantizar la entrega de mensajes.
* **Desventajas**:
  + **Requiere Conexión a Internet**: Necesita un broker MQTT accesible a través de Internet.
  + **No Es Seguro por Defecto**: Requiere cifrado adicional para seguridad.

**Conexión y Programación con ESP32 y MQTT**

**Conexión del ESP32 con un Broker MQTT**

1. **Configuración del Broker**: Puedes usar un broker MQTT público como **HiveMQ** o **Mosquitto**.

**Programación (Uso de la librería PubSubClient)**:

#include <WiFi.h>  
#include <PubSubClient.h>  
const char\* ssid = "your\_SSID";  
const char\* password = "your\_PASSWORD";  
const char\* mqtt\_server = "broker\_address";  
WiFiClient espClient;  
PubSubClient client(espClient);

void setup() {

Serial.begin(115200);  
 setup\_wifi();  
 client.setServer(mqtt\_server, 1883);}

void loop() {  
 if (!client.connected()) { reconnect(); }  
client.loop();

if (millis() - lastMsg > 5000) {  
lastMsg = millis();  
String msg = "Hello World";  
client.publish("topic/test", msg.c\_str());

}}  
void setup\_wifi() {

delay(10);  
Serial.println();  
Serial.print("Connecting to ");  
Serial.println(ssid);  
WiFi.begin(ssid, password);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("Connected!");

}

void reconnect() {

while (!client.connected()) {

Serial.print("Attempting MQTT connection...");  
 if (client.connect("ESP32Client")) {  
 Serial.println("connected");  
 client.subscribe("topic/test");

} else {  
Serial.print("failed, rc=");  
Serial.print(client.state());  
Serial.println(" try again in 5 seconds");  
delay(5000);

}

}

}

**Comparación   
LoRa**

* **Modelos**: Semtech SX1276, Dragino LoRa Shield, HopeRF RFM95W.
* **Aplicaciones**: Ideal para comunicación de largo alcance y bajo consumo energético en entornos extensos.
* **Ventajas**: Extenso alcance, bajo consumo.
* **Desventajas**: Ancho de banda limitado, latencia.

**MQTT**

* **Aplicaciones**: Ideal para la comunicación eficiente entre dispositivos IoT a través de Internet.
* **Ventajas**: Ligero, eficiente, escalable.
* **Desventajas**: Requiere conexión a Internet, seguridad adicional necesaria.

Estas soluciones te permitirán implementar un sistema de comunicación robusto y eficiente para tu proyecto IoT, adaptado a las necesidades específicas de comunicación de largo alcance o de red basada en Internet.