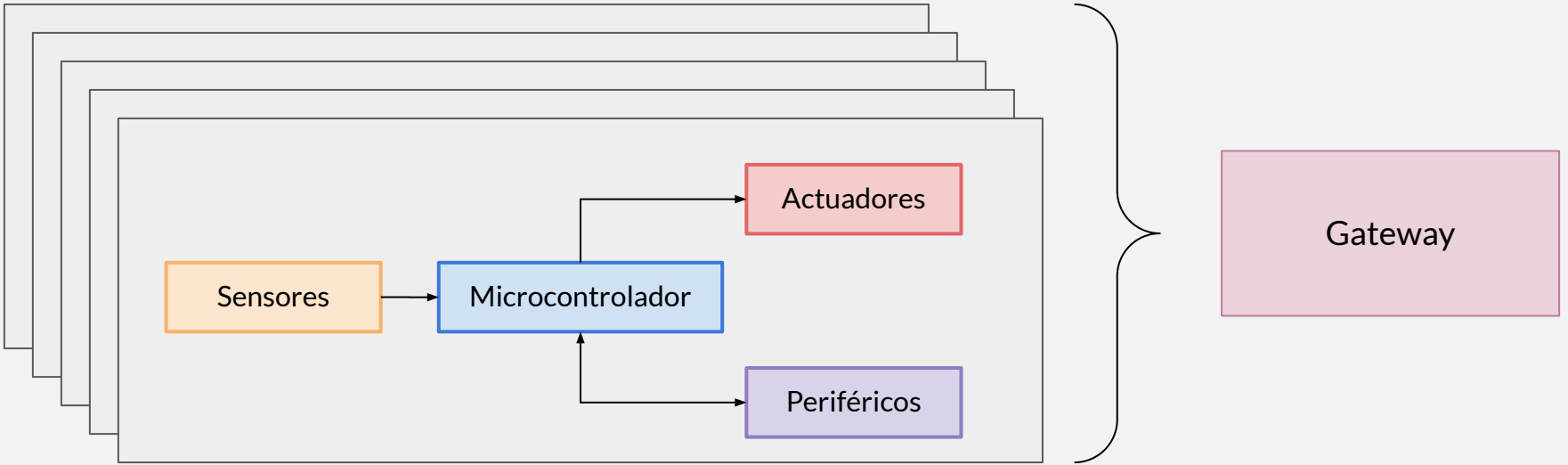
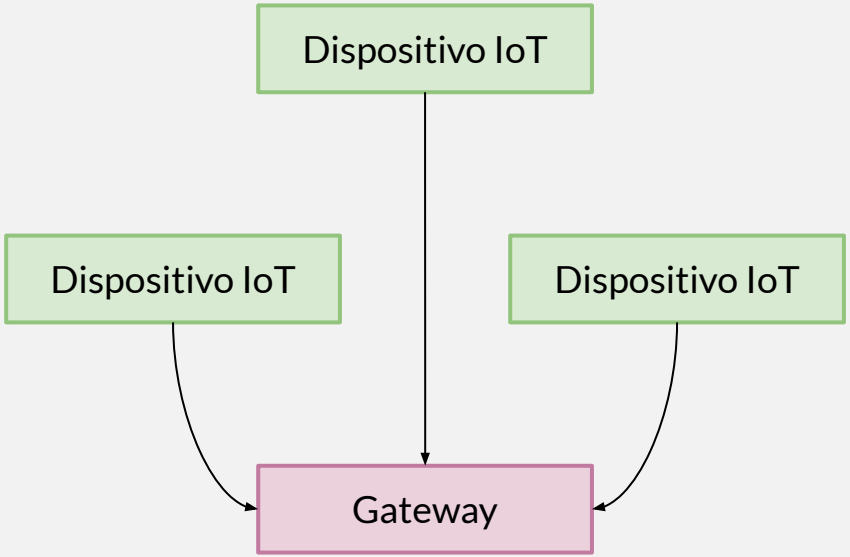
NEON
Network of Competence on Internet of Things
Co-funded by the Erasmus+ Programme of the European Union

Conectividad

Comunicación entre Dispositivos



Topología Estrella



Ventajas

- Nodos envían directamente a Gateway
- Gateway releva los datos y los procesa o transmite a la nube
- Nodos pueden *dormir* entre mensajes

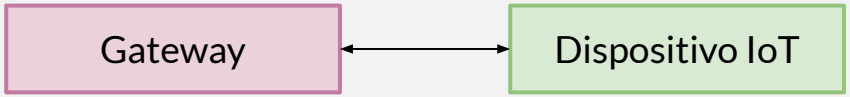
Desventajas

- Los nodos deben estar dentro del rango de acceso al Gateway

Tecnologías

- WiFi, LoRa, SigFox, Ethernet

Topología Punto a Punto



Ventajas

- Comunicación simple (unidireccional o bidireccional)

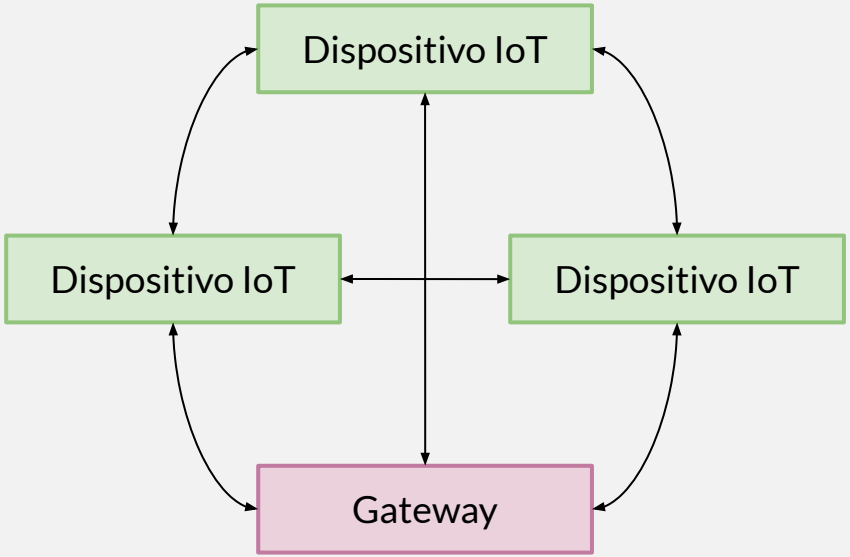
Desventajas

- Sólo permite conexión con un dispositivo a la vez

Tecnologías

- Bluetooth, Bluetooth Low Energy (BLE), Zigbee, RS232, Ethernet

Topología Malla



Ventajas

- Cada nodo envía a todos los nodos cercanos extendiendo la red
- Si falla un nodo, la red puede seguir funcionando

Desventajas

- Los dispositivos deben mantenerse activos para mantener el alcance de la red

Tecnologías

- Zigbee, WiFi Mesh, BLE Mesh

Tecnologías en ESP32 - WiFi

- **WiFi**
  - Adhiere a estándar IEEE 802.11 b/g/n
  - Permite modo estación y/o punto de acceso
  - Velocidad de transferencia de hasta 30Mb por UDP o 20Mb por TCP
  - Seguridad: WPA2/WPA3
- **WiFi Mesh (sólo ESP-IDF)**
  - Construido sobre protocolo WiFi
  - Cada nodo actúa como estación y punto de acceso
  - Cada nodo responde a un nodo padre o router
  - Cada nodo puede tener hijos asociados

Tecnologías en ESP32 - Bluetooth 4.2

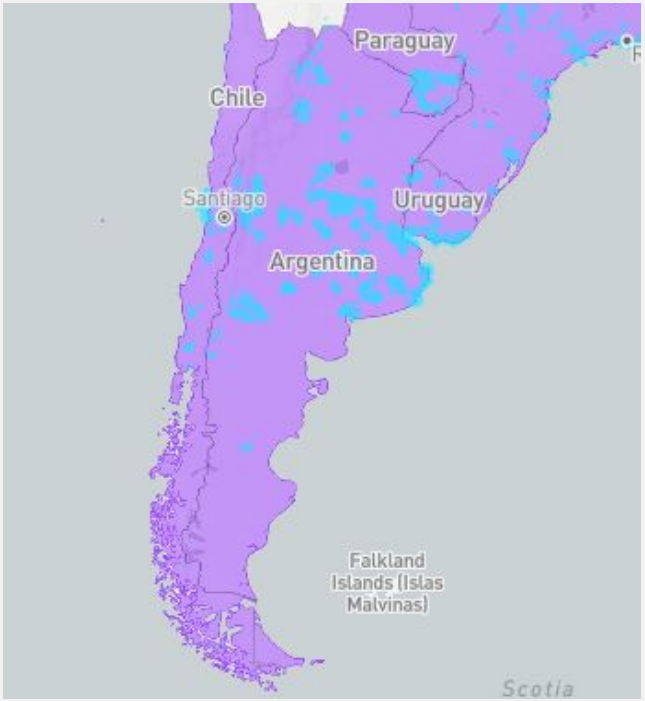
- **Bluetooth Clásico (sólo ESP-IDF)**
  - Banda 2.4 GHz
  - 79 canales de 1MHz
  - 1 Mb/s, 2 Mb/s, 3 Mb/s
- **Bluetooth Low Energy**
  - Banda 2.4 GHz
  - 40 canales de 2MHz
  - 125 Kb/s, 500 Kb/s, 1 Mb/s, 2 Mb/s
- **Bluetooth Low Energy Mesh (sólo ESP-IDF)**

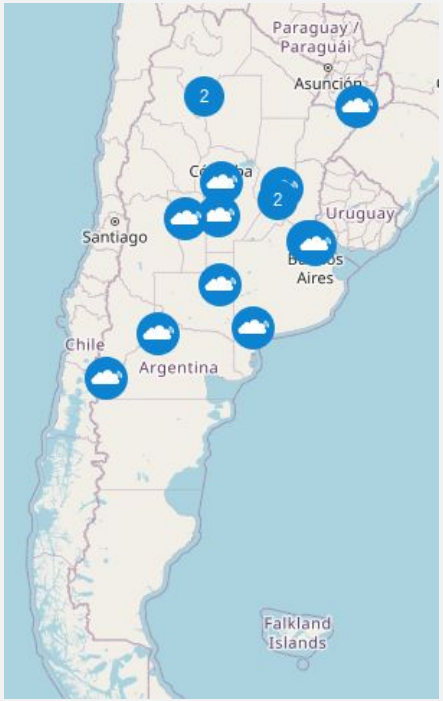
- **Zigbee**
  - Low Rate Wireless Area Network (IEEE 802.15.4)
  - Corto alcance
  - Baja potencia
  - Tasa de bits en el orden de kbps
  - Entradas y salidas disponibles

- **LoRa**
  - Tecnología propietaria
  - Largo alcance
  - Bajo consumo
  - Tasa de bits en el orden de bps
  - Programable por SPI

- **Sigfox**
  - Red inalámbrica global
  - Requiere licencia
  - Alto alcance
  - Bajo consumo

Cobertura Sigfox/LoRa



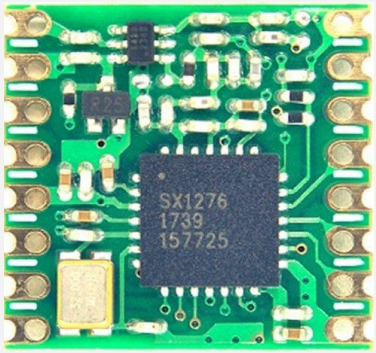


Ejemplo LoRa

Módulo LoRa

Especificaciones:

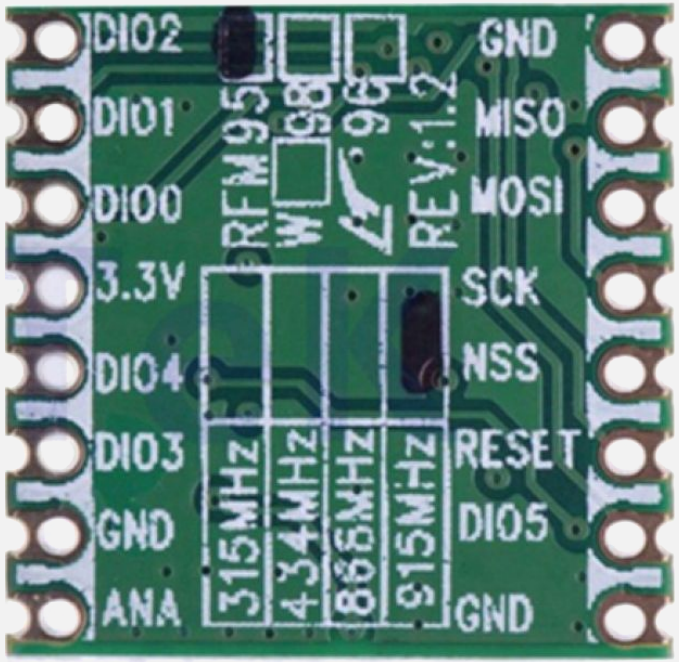
- Potencia emitida: 100mW
- Sensibilidad de recepción: hasta -148dBm
- Alimentación: 1.8V - 3.7V



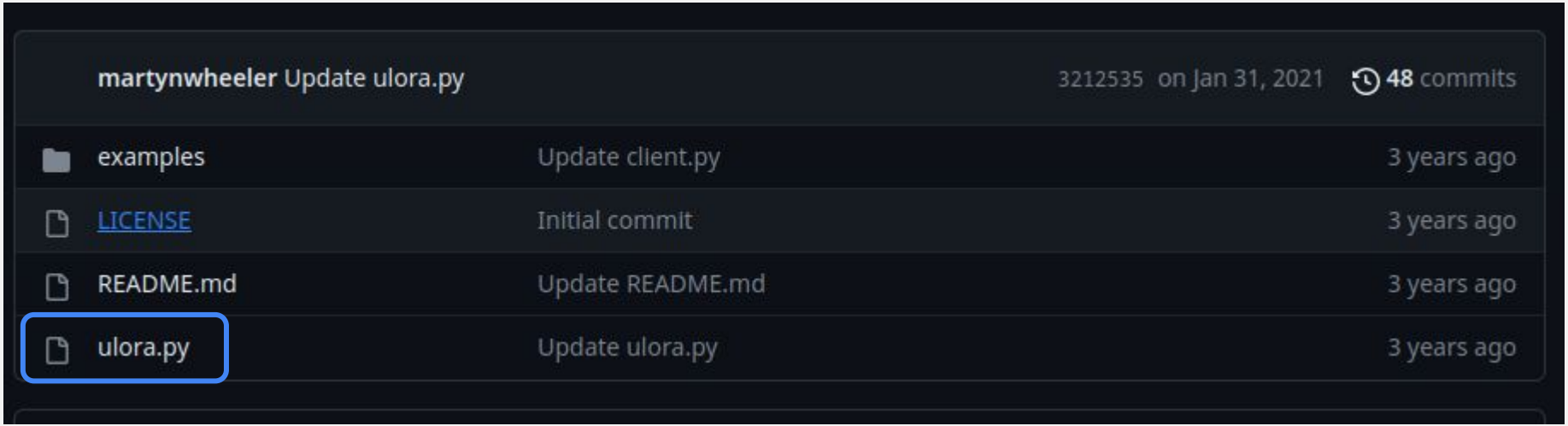
Semtech SX1276

Módulo LoRa - Conexiones

- Vcc
- GND
- SPI (MISO, MOSI, SCK/CLK, NSS/CS)
- RST: GPIO16
- DIO0 (IRQ): GPIO4



Módulo LoRa - Biblioteca



https://github.com/martynwheeler/u-lora

Módulo LoRa - Inicialización

```
from ulora import LoRa, SPIConfig

SPIBUS = SPIConfig.esp32_2
CS = 5; RST = 4; INT = 16
FREQ = 915.0; TX_POW = 20
DIR = 0x2

lora = LoRa(SPIBUS, INT, DIR, CS,
            reset_pin=RST, freq=FREQ,
            tx_power=TX_POW, acks=True)
```

Módulo LoRa - Inicialización

```
from ulora import LoRa, SPIConfig

SPIBUS = SPIConfig.esp32_2
CS = 5; RST = 4; INT = 16
FREQ = 915.0; TX_POW = 20
DIR = 0x2

lora = LoRa(SPIBUS, INT, DIR, CS,
            reset_pin=RST, freq=FREQ,
            tx_power=TX_POW, acks=True)
```

Módulo LoRa - Cliente/emisor

```
DIR_DEST = 0x1
n = 0
while True:
    n += 1
    if lora.send_to_wait(f"Mensaje de prueba {n}", DIR_DEST):
        print("Mensaje enviado")
    else:
        print("Falló transmisión del mensaje")

    sleep(2)
```

Módulo LoRa - Cliente/emisor

```
DIR_DEST = 0x1
n = 0
while True:
    n += 1
    if lora.send_to_wait(f"Mensaje de prueba {n}", DIR_DEST):
        print("Mensaje enviado")
    else:
        print("Falló transmisión del mensaje")

    sleep(2)
```

Módulo LoRa - Cliente/emisor

```
DIR_DEST = 0x1
n = 0
while True:
    n += 1
    if lora.send_to_wait(f"Mensaje de prueba {n}", DIR_DEST):
        print("Mensaje enviado")
    else:
        print("Falló transmisión del mensaje")

    sleep(2)
```

Módulo LoRa - Servidor/receptor

```
def on_recv(payload):
    print("Desde:", payload.header_from)
    print("Recibido:", payload.message)
    print("RSSI: {}; SNR: {}".format(payload.rssi, payload.snr))

lora.on_recv = on_recv
lora.set_mode_rx()
while True:
    sleep(0.1)
```

Módulo LoRa - Servidor/receptor

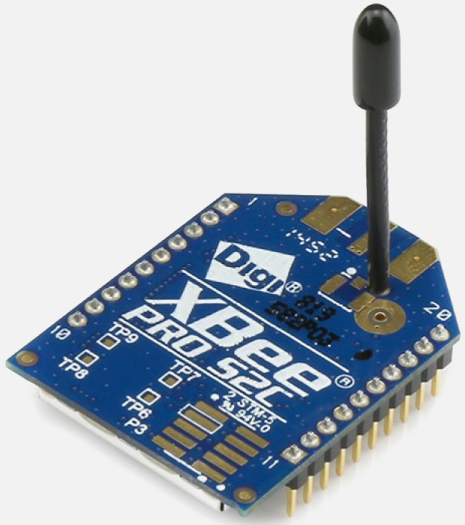
```
def on_recv(payload):
    print("Desde:", payload.header_from)
    print("Recibido:", payload.message)
    print("RSSI: {}; SNR: {}".format(payload.rssi, payload.snr))

lora.on_recv = on_recv
lora.set_mode_rx()
while True:
    sleep(0.1)
```

Ejemplo Zigbee

Módulo XBee PRO S2C

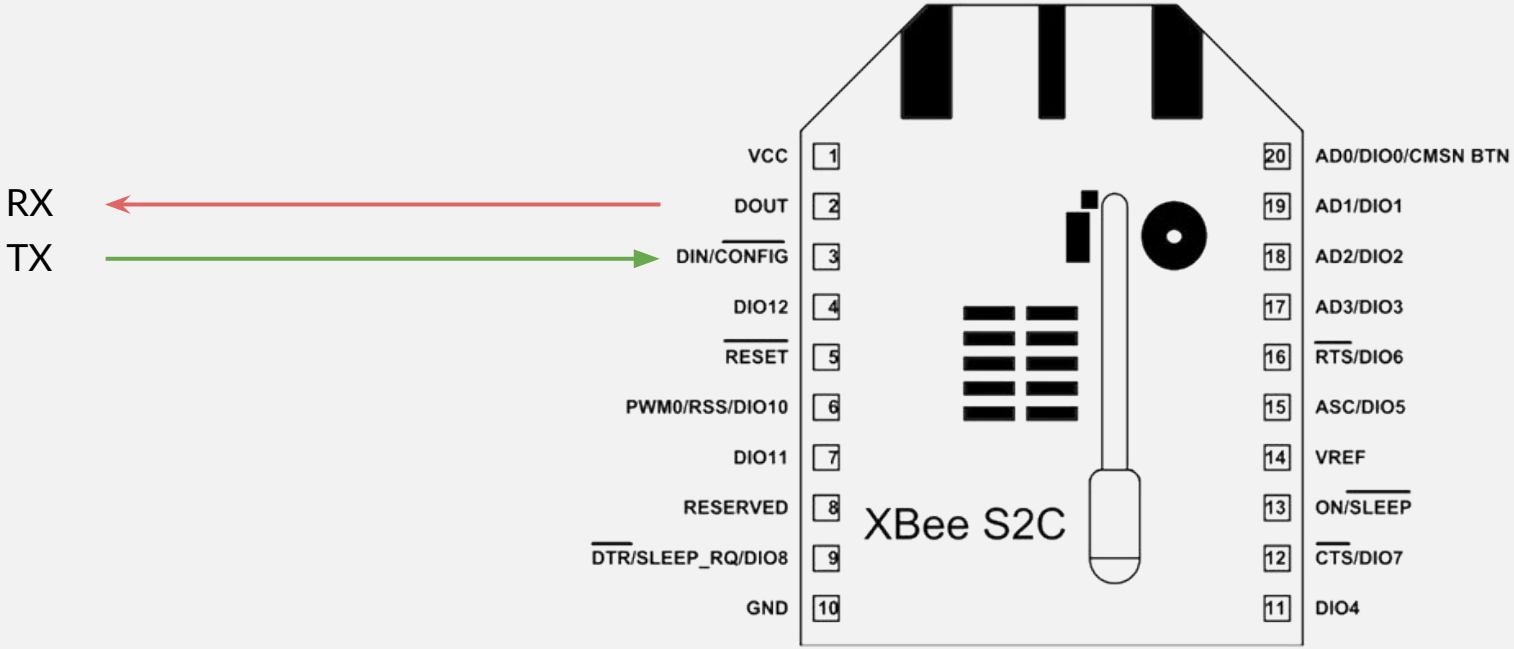
Especificaciones:

- Potencia emitida: 63 mW
- Sensibilidad de recepción: -101dBm
- Alimentación: 2.7V - 3.6V

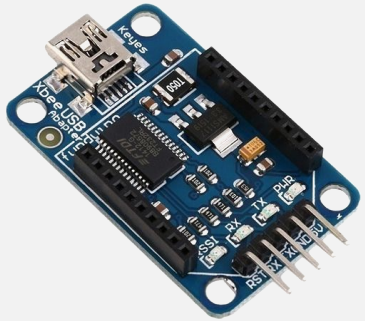


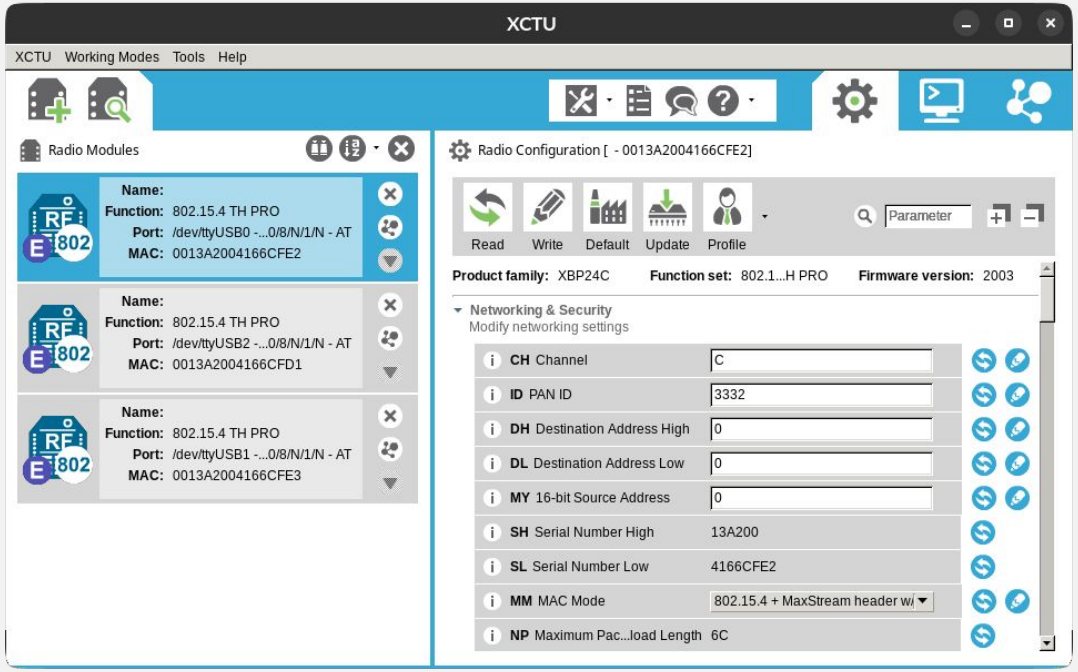
XBee PRO S2C

Módulo XBee PRO S2C

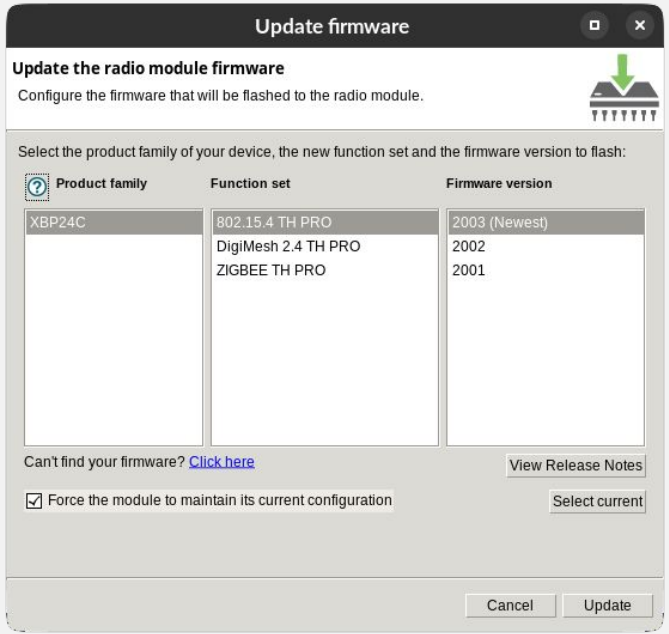


Configuración Módulo XBEE

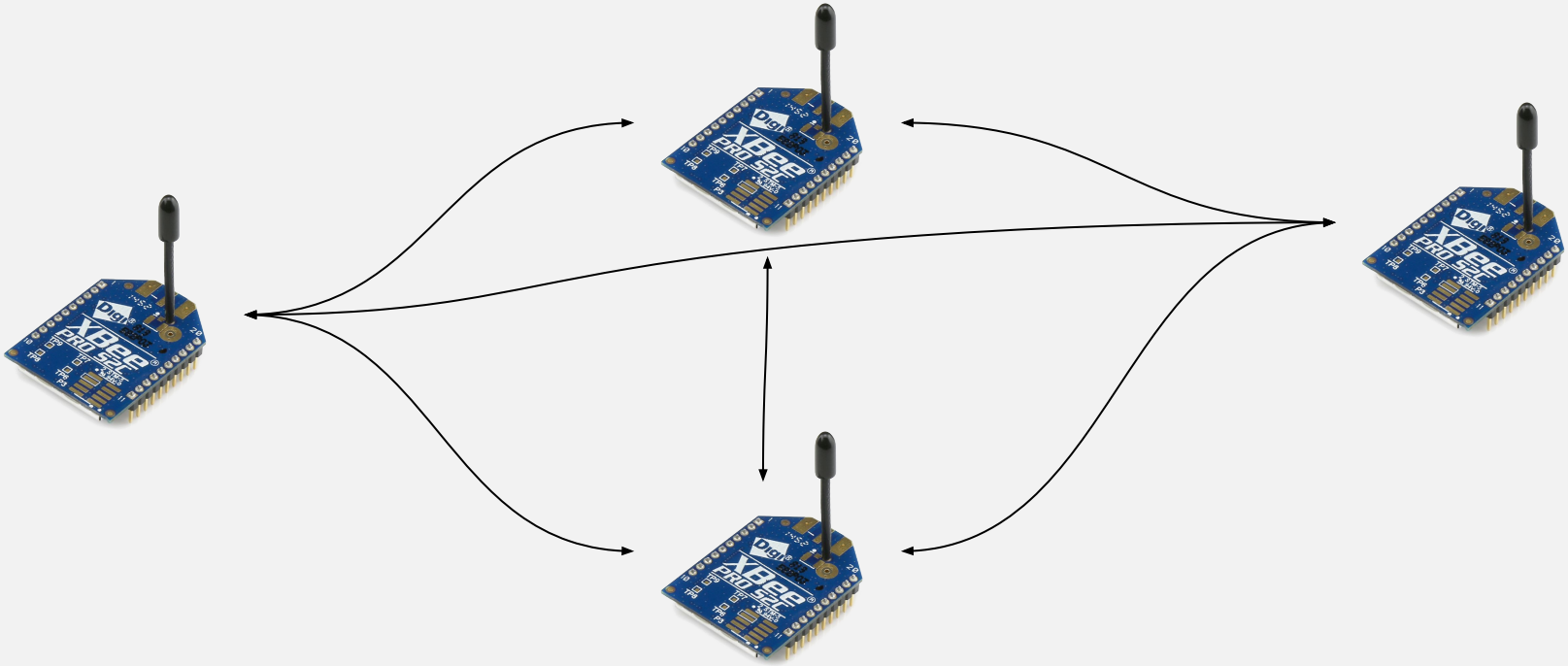




Configuración Módulo XBEE



Red ZigBee 802.15.4



Módulo UART MicroPython

```
from machine import UART

uart = UART(2, baudrate=9600)

while True:
    if uart.any() > 0:
        buf = uart.read()
        print(buf.decode(), end="")
```

Pines asignados para cada puerto UART

	UART0 (USB)	UART1 (FLASH)	UART2
tx	1	10	17
rx	3	9	16

Módulo UART MicroPython

```
from machine import UART
from time import sleep_ms

uart = UART(2, baudrate=9600)
n = 0

while True:
    n += 1
    uart.write(f"Mensaje de prueba {n}")
    sleep_ms(1000)
```

Pines asignados para cada puerto UART

	UART0 (USB)	UART1 (FLASH)	UART2
tx	1	10	17
rx	3	9	16

Ejemplo BLE

BLE - Roles

- Broadcaster
  - Anuncia periódicamente datos a todos los dispositivos cercanos
- Observer
  - Escanea anuncios periódicos de otros dispositivos para ver cuáles están en cercanía
- Peripheral
  - Dispositivo de bajo consumo que emite información a un nodo central. Solo puede estar conectado a un central en simultáneo
- Central
  - Se puede conectar a uno o más periféricos

Ejemplo BLE UART

Requiere ble_simple_peripheral.py y ble_advertising.py que proveen los ejemplos de MicroPython (disponible en campus)

```
1  import bluetooth
2  from time import sleep_ms
3
4  from ble_simple_peripheral import BLESimplePeripheral
5
6  ble = bluetooth.BLE()
7  p = BLESimplePeripheral(ble)
```

Declaración de interfaz BLE y periférico

Ejemplo BLE UART

```
 9  def on_rx(v):
10      print("RX", v)
11
12  p.on_write(on_rx)
13
14  while True:
15      if p.is_connected():
16          p.send("mensaje")
17      sleep_ms(100)
```

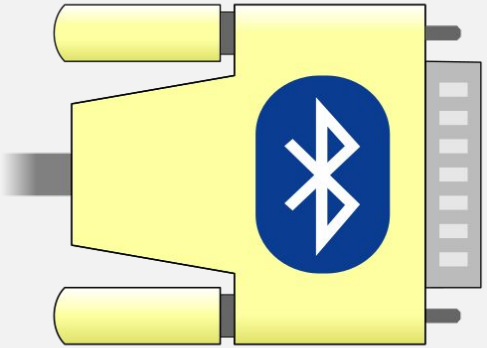
Definición del callback de recepción de mensaje

Envío mensajes periódicamente

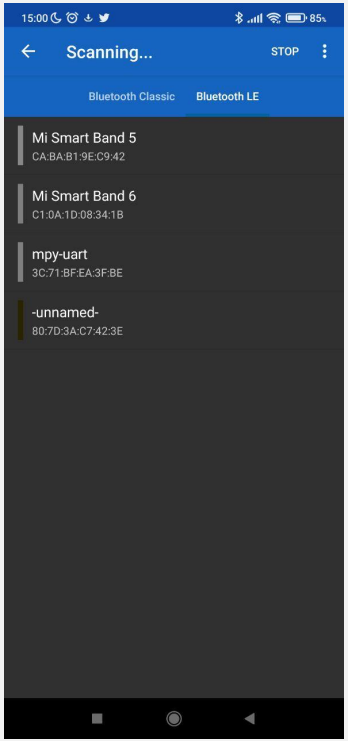
Ejemplo BLE UART

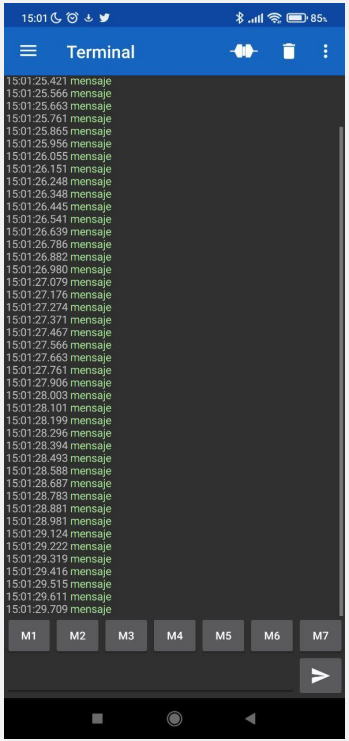
Una vez ejecutado el programa, se puede conectar un dispositivo central (celular Android) para interactuar con el dispositivo

Serial Bluetooth Terminal



Ejemplo BLE UART - En Android





Ejemplo BLE UART - En Thonny