# LanuzaConnect

Manual de Uso

LanuzaConnect es una biblioteca para ESP32 diseñada para facilitar y simplificar el manejo de las funciones de comunicación entre dos placas ESP32 proporcionadas por ESP32-NOW. En este documento se puede ver un resumen del uso de esta biblioteca.

### La biblioteca

La biblioteca está contenida dentro de una carpeta llamada LanuzaConnect. Dentro de esta carpeta hay dos carpetas: *examples* y *src*. La primera contiene ejemplos para la biblioteca. En la segunda se encuentra el código de la biblioteca, en los archivos *LanuzaConnect.h* y *LanuzaConnect.cpp*. Además, la biblioteca contiene una archivo *library.properties*, con información sobre la biblioteca y este manual en formato PDF.

### Instalación

Dado que la biblioteca no está publicada, hay que instalarla desde un archivo .zip. Para ello, sigue los siguientes pasos:

1. Descarga el archivo **LanuzaConnect.zip** a tu carpeta de descargas habitual.
2. Abre Arduino IDE, y, haz clic en Sketch > Incluir Biblioteca > Añadir Biblioteca .ZIP
3. Selecciona el archivo descargado previamente.
4. La biblioteca estará instalada. En la consola debería aparecer el mensaje "**Librería instalada**".

### Uso de la biblioteca

Aquí hay un ejemplo del código que habría que incluir para usar la librería:

```
#include <LanuzaConnect.h>

LanuzaConnect esp("TRANSMITTER");

// Dirección MAC del receptor
uint8_t receiver[] = {0xB8, 0xD6, 0x1A, 0x47, 0xC3, 0x3C};

void setup() {
    Serial.begin(115200);
    esp.communicationStartup();
    esp.addPeer(receiver);

}

void loop() {
```

```
    esp.sendString("Hola Mundo", receiver);
    delay(5000);
}
```

**Funciones de la biblioteca**

NOTA: Todas las funciones se aplican a un objeto de la clase LanuzaConnect, como esp en el código anterior (ej. **esp**.communicationStartup();)

### `LanuzaConnect esp(mode);`

Crea un objeto de la clase LanuzaConnect para ser usado en todas la funciones posteriores. En el parámetro *mode* se puede poner “TRANSMITTER” para un transmisor, “RECEIVER” para un receptor o “TRANSCEIVER” para un transmisor-receptor.

### `uint8_t receiver[] = {mac};`

Esto no es una función como tal, sino un ejemplo de la creación de una variable para almacenar la dirección MAC del receptor, que hay que introducir en el parámetro mac. Esto tiene que tener el formato de 6 bytes en hexadecimal separados por comas cada uno con 0x al principio, como por ejemplo 0xB8, 0xD6, 0x1A, 0x47, 0xC3, 0x3C en el código anterior.

### `esp.communicationStartup();`

Esta función sirve para iniciar la comunicación. Ha de haberse definido el objeto primero. Al usarla debería aparecer en el puerto serie el texto siguiente:

```
MAC Address: B0:A7:32:16:1F:9C (la MAC de la placa en cuestión)
WiFi initialized
ESP-NOW initialized
Mode: Transmitter (o el modo correspondiente)
```

### `esp.addPeer(receiver);`

Esta función añade al receptor. En el parámetro receiver, hay que introducir la MAC del receptor, en el formato especificado antes. Aunque se podría introducir aquí la MAC directamente, es recomendable crear una constante con su valor como se ha hecho antes y usarla aquí. Al usarla debería aparecer en el puerto serie el texto siguiente:

```
Peer added: B8:D6:1A:47:C3:3C (la MAC del receptor)
```

### `esp.sendString(string, receiver);`

Esta función envía una cadena de texto al receptor especificado. En el parámetro string hay que introducir la cadena de texto, entre comillas, y en el parámetro receiver la MAC del receptor. Al ejecutarla debería aparecer lo siguiente en el puerto serie:

```
Sent successfully data: Hola Mundo (o la string correspondiente)
Text sent; Delivery Success (si sale Fail no se ha recibido)
```

### `esp.OnDataRecv();`

Esta función no es necesario ejecutarla, ya que se ejecuta por sí sola sin que aparezca en el código cada vez que una placa configurada como receptor recibe algo. Al recibir datos, debería mostrarse el siguiente texto por el puerto serie:

```
Bytes received: 250 (aunque no sea el número real)
String received: Hola Mundo (o la string que toque)
```

**Usos avanzados:**

Es posible usar la librería para enviar texto a dos receptores diferentes. Para ello, en el emisor hay que cargar el ejemplo *multitransmitter.ino* de la biblioteca, que crea dos constantes diferentes con dos direcciones de dos receptores (puede ser ampliado a más). Luego, simplemente hay que añadir a los dos como peers y seleccionar el correcto al enviar. Los receptores pueden seguir usando el código de receptor.