Communication LR entre 2 ESP32

Librairie à installer :

WiFi

Built-In by **Arduino** Version **1.2.7** **INSTALLED**
Enables network connection (local and Internet) using the Arduino WiFi shield. With this library you can instantiate Servers, Clients and send/receive UDP packets through WiFi. The shield can connect either to open or encrypted networks (WEP, WPA). The IP address can be assigned statically or through a DHCP. The library can also manage DNS.
More info

Adafruit BME280 Library

by **Adafruit** Version **2.2.1** **INSTALLED**
Arduino library for BME280 sensors. Arduino library for BME280 humidity and pressure sensors.
More info

Adafruit BusIO

by **Adafruit** Version **1.9.3** **INSTALLED**
This is a library for abstracting away UART, I2C and SPI interfacing This is a library for abstracting away UART, I2C and SPI interfacing
More info

Adafruit Circuit Playground

Built-In by **Adafruit** Version **1.11.3** **INSTALLED**
All in one library to control Adafruit's Circuit Playground board. All in one library to control Adafruit's Circuit Playground board.
More info

Adafruit GFX Library

by **Adafruit** Version **1.10.12** **INSTALLED**
Adafruit GFX graphics core library, this is the 'core' class that all our other graphics libraries derive from. Install this library in addition to the display library for your hardware.
More info

Adafruit SSD1306

by **Adafruit** Version **2.5.0** **INSTALLED**
SSD1306 oled driver library for monochrome 128x64 and 128x32 displays SSD1306 oled driver library for monochrome 128x64 and 128x32 displays
More info

Adafruit Unified Sensor

by **Adafruit** Version **1.1.4** **INSTALLED**
Required for all Adafruit Unified Sensor based libraries. A unified sensor abstraction layer used by many Adafruit sensor libraries.
More info

Sélectionner une version | Installer

Gestionnaire de carte :

esp32

by **Espressif Systems** version **1.0.6** **INSTALLED**
Cartes incluses dans ce paquet:
ESP32 Dev Module, WEMOS LoLin32, WEMOS D1 MINI ESP32.
More Info

Arduino AVR Boards

Built-In by **Arduino** version **1.8.3** **INSTALLED**
Cartes incluses dans ce paquet:
Arduino Yún, Arduino Uno, Arduino Uno Mini, Arduino Uno WiFi, Arduino Diecimila, Arduino Nano, Arduino Mega, Arduino MegaADK, Arduino Leonardo, Arduino Leonardo Ethernet, Arduino Micro, Arduino Esplora, Arduino Mini, Arduino Ethernet, Arduino Fio, Arduino BT, Arduino LilyPadUSB, Arduino Lilypad, Arduino Pro, Arduino ATMegaNG, Arduino Robot Control, Arduino Robot Motor, Arduino Gemma, Adafruit Circuit Playground, Arduino Yún Mini, Arduino Industrial 101, Linino One.
Online Help
More Info

Sélectionner une version | Installer | Mise à jour

Pour une communication entre 2 esp32 il faut établir un émetteur et un récepteur. Il faut que l'émetteur connaisse l'adresse MAC du récepteur pour cela, il faut utiliser le programme ci-dessous pour la récupérer :

```
#include "WiFi.h"

void setup(){
  Serial.begin(115200);
  WiFi.mode(WIFI_MODE_STA);
  Serial.println(WiFi.macAddress());
}

void loop(){

}
```

Lors du lancement du programme, il se peut que Arduino affiche une erreur il suffit de relancer le programme et d'appuyer sur le bouton boot longtemps le temps que Arduino lance les % d'écritures.

Suite à ça, il faut écrire un code émetteur et un code récepteur en mettant bien l'adresse mac trouvé dans le code émetteur.

Récepteur :

```
//Recepteur
#include "esp_now.h"
#include "WiFi.h"

uint8_t HELTECAdresse []={0x24, 0x0A, 0xC4, 0x0D, 0x8D, 68};

void setup(){
  // Init Serial Monitor
  Serial.begin(115200);

  // Set ESP32 as a WIFI Station
  WiFi.mode(WIFI_STA);

  // Init ESP-NOW
  esp_now_init();

  //Fonction callback
  esp_now_register_recv_cb(OnDataRecv);

}

typedef struct struct_message
{
  char a[32];
  int b;
  float c;
  String d;
```

```
  bool e;
}struct_message;

struct_message myData;



void OnDataRecv(const uint8_t *mac, const uint8_t *incomingData, int len){
  memcpy(&myData, incomingData, sizeof(myData));
  Serial.println("Donnees recues");
}



void loop(){
  Serial.print("b : ");
  Serial.print(myData.b,DEC);
  Serial.print("\t");

}
```

Emetteur :

```
//Emetteur
#include "esp_now.h"
#include "WiFi.h"

uint8_t HELTECAdresse []={0x24, 0x0A, 0xC4, 0x0D, 0x8D, 68};

void setup(){
 // Init Serial Monitor
 Serial.begin(115200);

 // Set ESP32 as a WIFI Station
 WiFi.mode(WIFI_STA);

 // Init ESP-NOW
 if(esp_now_init()==ESP_OK)
 {
  Serial.println("Initialisation ESP OK");
 }
 else{
  Serial.println("Erreur initialisation ESP");
  ESP.restart();
 }
 //Fonction callback
 esp_now_register_send_cb(OnDataSent);

 //Strucute Appairage INFO
 struct esp_now_peer_info peerInfo;
 memcpy(peerInfo.peer_addr, HELTECAdresse, 6);
```

```
  peerInfo.channel=0;
  peerInfo.encrypt=false;

  // test
  if(esp_now_add_peer(&peerInfo)==ESP_OK)
  {Serial.println("Fonction peer OK");}
  else
  {Serial.println("Erreur fonction peer");}



}

//Information sur l'envoi de la data
void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status){
  if(status == ESP_NOW_SEND_SUCCESS)
  {
    Serial.println("livraison OK");}
    else{Serial.println("OK");}
}

typedef struct struct_message
{
  char a[32];
  int b;
  float c;
  String d;
  bool e;
}struct_message;
```

```
struct_message myData;


void loop(){
   strcpy(myData.a,"SALUT TWAAA");
  myData.b = random(1,20);
  myData.c = 1.2;
  myData.d = "HELLO";
  myData.e= false;


  esp_err_t result=esp_now_send(HELTECAdresse,(uint8_t*)&myData,sizeof(myData));
  if (result==ESP_OK)
    {Serial.println("Envoi OK");}
    else{Serial.println("erreur Envoi");}


  delay(500);
}
```

Voila les programmes qu'on à effectuer qui ne fonctionne pas complétement, cependant on arrive à écrire sur le moniteur série, il ne doit pas manquer grands choses pour avoir la communication entre les 2 ESP32.

Après avoir réussi à faire communiquer les ESP32, il faudrait suivre la partie qui suit pour effectuer la communication LR :

Partie Communication LR :

La communication LR va avec le mode 802.11, ce sont des modes de protocoles wifi :

Protocol Mode	Description
802.11b	Call esp_wifi_set_protocol(ifx, WIFI_PROTOCOL_11B) to set the station/AP to 802.11b-only mode.
802.11bg	Call esp_wifi_set_protocol(ifx, WIFI_PROTOCOL_11B|WIFI_PROTOCOL_11G) to set the station/AP to 802.11bg mode.
802.11bgn	Call esp_wifi_set_protocol(ifx, WIFI_PROTOCOL_11B| WIFI_PROTOCOL_11G|WIFI_PROTOCOL_11N) to set the station/ AP to BGN mode.
802.11 BGNLR	Call esp_wifi_set_protocol(ifx, WIFI_PROTOCOL_11B| WIFI_PROTOCOL_11G|WIFI_PROTOCOL_11N|WIFI_PROTOCOL_LR) to set the station/AP to BGN and the Espressif-specific mode.
802.11 LR	Call esp_wifi_set_protocol(ifx, WIFI_PROTOCOL_LR) to set the station/AP only to the Espressif-specific mode. **This mode is an Espressif-patented mode which can achieve a one-kilometer line of sight range. Please, make sure both the station and the AP are connected to an ESP device.**

Le mode Longue Portée (LR) est un mode Wi-Fi unique d'Espressif cela veut dire que seul les appareils ESP32 peuvent transmettre et recevoir les données. De plus, le mode peut atteindre une portée d'environ 1 Km.

Le **débit LR** a un débit très limité car le débit de données PHY brut LR est de ½ Mbits et ¼ Mbits.

Pour utiliser LR :

- Le point d'accès et la station sont tous deux des appareils.
- Une connexion Wi-Fi longue distance et une transmission de données sont nécessaires
- Les exigences de débit de données sont très faibles, telles que le contrôle a distance des périphériques, etc

L’activation du mode est un simple appel de fonction pour basculer votre ESP32 en mode LR voici un exemple :

```
void init_wifi(wifi_mode_t mode)
{
   const uint8_t protocol = WIFI_PROTOCOL_LR;
   tcpip_adapter_init();
   ESP_ERROR_CHECK( esp_event_loop_init(event_handler, NULL) );
   wifi_init_config_t cfg = WIFI_INIT_CONFIG_DEFAULT();
   ESP_ERROR_CHECK( esp_wifi_init(&cfg) );
   ESP_ERROR_CHECK( esp_wifi_set_storage(WIFI_STORAGE_RAM) );
   ESP_ERROR_CHECK( esp_wifi_set_mode(mode) );
   wifi_event_group = xEventGroupCreate();

   if (mode == WIFI_MODE_STA) {
      ESP_ERROR_CHECK( esp_wifi_set_protocol(WIFI_IF_STA, protocol) );
      wifi_config_t config = {
         .sta = {
            .ssid = ap_name,
            .password = pass,
            .bssid_set = false
         }
      };
      ESP_ERROR_CHECK( esp_wifi_set_config(WIFI_IF_STA, &config) );
      ESP_ERROR_CHECK( esp_wifi_start() );
      ESP_ERROR_CHECK( esp_wifi_connect() );

      xEventGroupWaitBits(wifi_event_group, CONNECTED_BIT,
                  false, true, portMAX_DELAY);
      ESP_LOGI(TAG, "Connected to AP");
```

```
    } else {
        ESP_ERROR_CHECK( esp_wifi_set_protocol(WIFI_IF_AP, protocol) );
        wifi_config_t config = {
            .ap = {
                .ssid = ap_name,
                .password = pass,
                .ssid_len = 0,
                .authmode = WIFI_AUTH_WPA_WPA2_PSK,
                .ssid_hidden = false,
                .max_connection = 3,
                .beacon_interval = 100,
            }
        };
        ESP_ERROR_CHECK( esp_wifi_set_config(WIFI_IF_AP, &config) );
        ESP_ERROR_CHECK( esp_wifi_start() );
    }
}
```