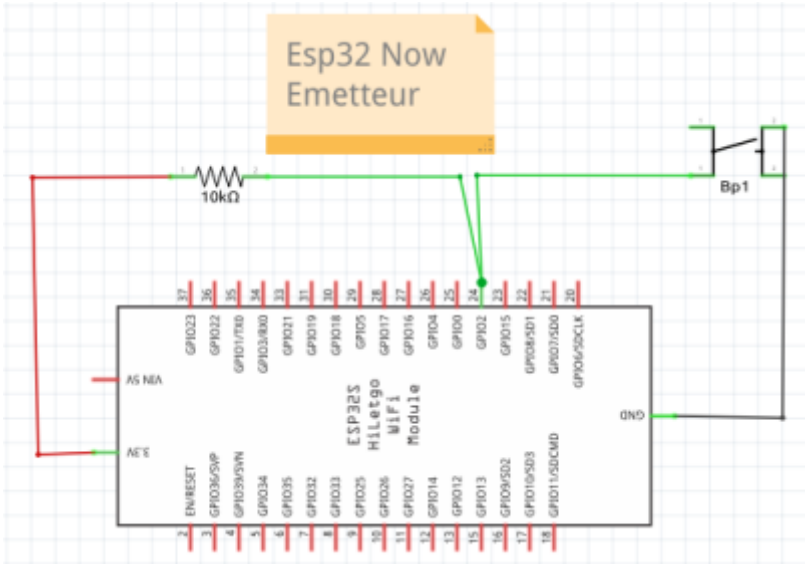
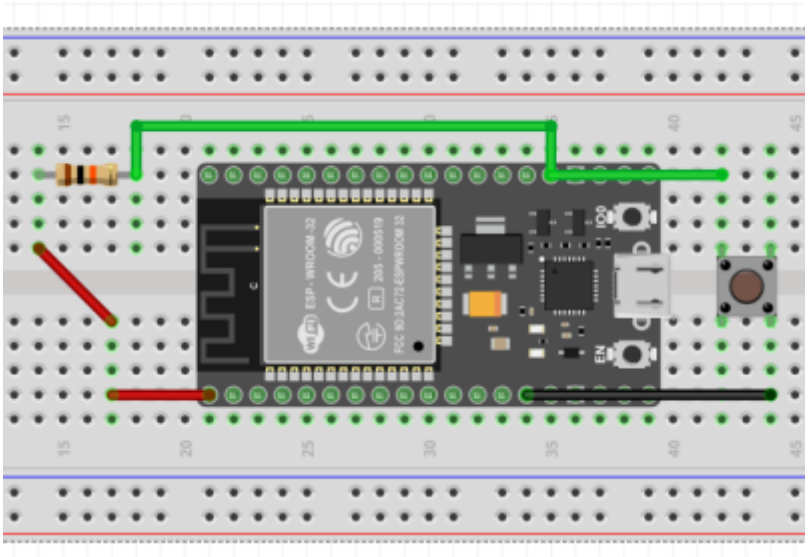
Travaux pratique Esp32 Now

Emetteur avec Bouton poussoir

Schema de raccordement





emetteurBP.ino

```
/*
  Raccorder un Bouton poussoir sur la broche 34 ( pour essai)
  Et remplacer l'adresse MAC par @Mac de votre autre ESP32
  il faut appuyer environ 1 s sur le Bp1
*/
#include <esp_now.h>
#include <WiFi.h>
```

```
const int Bp1  = 34;
int MemLed1 = 0;
int MemBp1 = 0;
int tempo = 50;
String result = " ";



// Indiquer   l'MAC Address de l 'ESP32 distant
uint8_t broadcastAddress[] = {0x58, 0xbf, 0x25, 0x14, 0x24, 0x34};

// Structure d'envoi du message
typedef struct struct_message {
  char a[32];
  int b;
  float c;
  bool d;
} struct_message;

// Creation d une structure de message appellée  myData
struct_message myData;




esp_now_peer_info_t peerInfo;

// Retour de message de status  l'autre esp
void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status)
{
  Serial.print("\r\nLast Packet Send Status:\t");
  Serial.println(status == ESP_NOW_SEND_SUCCESS ? "Bouton OK" : "Erreur
Bouton");
}

void setup() {

  Serial.begin(115200);

  pinMode(Bp1, INPUT);


  WiFi.mode(WIFI_STA);

  // Initiation  ESP-NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;
```

```
  }


  //Recupere  le status du paquet envoyer
  esp_now_register_send_cb(OnDataSent);

  // test l'adress Mac , Envoi sur canal 0, et ne pas crypter le paquer
envoyer
  memcpy(peerInfo.peer_addr, broadcastAddress, 6);
  peerInfo.channel = 0;
  peerInfo.encrypt = false;


  if (esp_now_add_peer(&peerInfo) != ESP_OK){
    Serial.println("Failed to add peer");
    return;
  }

}

void loop() {
  int valeurBp1 = digitalRead(Bp1);// On lit la valeur de Bp1 au début
de la boucle
  delay(tempo);

  strcpy(myData.a, " data1");
  myData.b = 0;
  myData.c = 0.5;

   //Temps  1
  if (valeurBp1 == 1 && MemBp1 ==0 && MemLed1 == 0) {
       myData.d = 1;;MemBp1 =1;MemLed1 = 1;
       esp_err_t result = esp_now_send(broadcastAddress, (uint8_t *)
&myData, sizeof(myData));
      }
  //Temps 2
  if (valeurBp1 == 0 && MemBp1 ==1 && MemLed1 == 1) {
       myData.d = 1;;MemBp1 =0;MemLed1 = 1;
      }
  //Temps 3
  if (valeurBp1 == 1 && MemBp1 ==0 && MemLed1 == 1) {
       myData.d = 0;;MemBp1 =1;MemLed1 = 0;
      }
  //Temps 4
  if (valeurBp1 == 0 && MemBp1 ==1 && MemLed1 == 0) {
       myData.d = 0;;MemBp1 =0;MemLed1 = 0;
       esp_err_t result = esp_now_send(broadcastAddress, (uint8_t *)
&myData, sizeof(myData));
      }

  Serial.println(myData.d);// affiche la valeur envoyée mydata.d ==>
```

```
booleen

  if (result == ESP_OK) {
    Serial.println("Sent with success");
  }
  if (result == " " ){
    Serial.println("Attente d'appui !!!");
  }
  else {
    Serial.println("Error sending the data");
  }
  delay(1000);

}
```

Recepteur avec Led

recepteur001.ino

```
/*
  Raccorder la led sur la sortie 2
*/

#include <esp_now.h>
#include <WiFi.h>

int led1 = 2;
int tempo = 50;

// Structure example to receive data
// Must match the sender structure
typedef struct struct_message {
    char a[32];
    int b;
    float c;
    bool d;
} struct_message;

// Create a struct_message called myData
struct_message myData;

// callback function that will be executed when data is received
void OnDataRecv(const uint8_t * mac, const uint8_t *incomingData, int
len) {
  memcpy(&myData, incomingData, sizeof(myData));
  Serial.print("Bytes received: ");
  Serial.println(len);
```

```
  Serial.print("Char: ");
  Serial.println(myData.a);
  Serial.print("Int: ");
  Serial.println(myData.b);
  Serial.print("Float: ");
  Serial.println(myData.c);
  Serial.print("Bool: ");
  Serial.println(myData.d);
  Serial.println();
}

void setup() {
  // Initialize Serial Monitor
  Serial.begin(115200);
  pinMode(led1, OUTPUT);

  // Set device as a Wi-Fi Station
  WiFi.mode(WIFI_STA);

  // Init ESP-NOW
  if (esp_now_init() != ESP_OK) {
    Serial.println("Error initializing ESP-NOW");
    return;
  }

}

void loop() {
  esp_now_register_recv_cb(esp_now_recv_cb_t(OnDataRecv));
  delay(1000);

if (myData.d == true) {
  digitalWrite(led1, HIGH);
  }
if (myData.d == false) {
  digitalWrite(led1, LOW);
  }
}
```