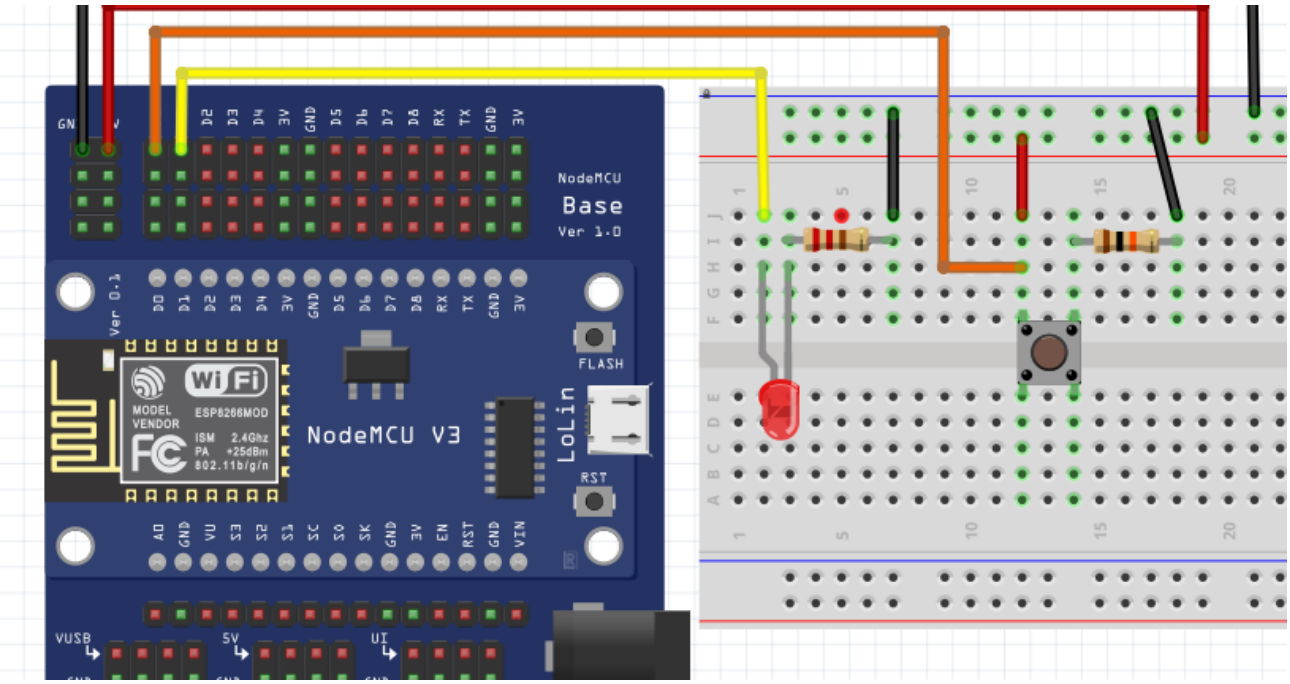
NodeMCU: 07_ESPNOW (Comunicación bidireccional) ¡¡EUREKA!!

agrportfolioeducativo.blogspot.com/2020/03/nodemcu-07espnow-comunicacion.html

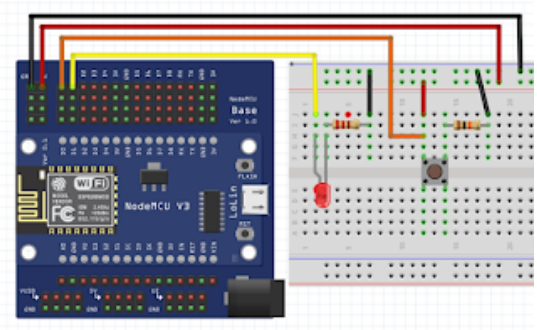


Entro ahora en terreno desconocido. No he leído nada, por ahora, que me permita hacer una comunicación ESP-NOW de forma bidireccional con los ESP8266, aunque puedo obtener pistas de la entrada NodeMCU: 04_ESPNOW (Comunicación maestro-esclavo).

La idea y paso previo

La idea es conectar un NodeMCU a un LED en D1 y a un pulsador en D0. Y asimismo otro segundo NodeMCU. Cuando pulse el pulsador del primer NodeMCU debe iluminarse el LED del segundo, y viceversa. Vamos, este circuito para ambos:

Para probar que está todo bien conectado... Así, si pulsamos el botón, se enciende el LED del NodeMCU.



```
int entrada = D0;
int salida = D1;

void setup() {
pinMode(entrada,INPUT);
pinMode(salida,OUTPUT);
}

void loop() {
digitalWrite(salida,digitalRead(entrada));

}
```

view raw NODEMCU_AP+STATION_previo.ino hosted with ❤ by GitHub

Y ahora el desafío: al pulsar el pulsador, se debe encender el LED del ***otro NodeMCU***...

= = = = = = = = = = =

El programa del primer NodeMCU

En este programa tenemos que hacer dos cosas (respecto a comunicaciones master-slave ESPNOW anteriores), que son:

1. Utilizar el role 3 --> esp_now_set_self_role(3);
2. Utilizar la MAC de la otra estación en la línea de emparejamiento (esp_now_add_peer) y en el envío de datos (esp_now_send).
3. Utilizar el callback de envío (esp_now_register_send_cb) y también el de recepción (esp_now_register_recv_cb), y en este último, programar los resultados. En mi caso, encender un LED.

Una vez comprendido el sistema, podremos hacer maravillas conectando ambos dispositivos. Habrá que ver cómo hacer para conectar más de uno...

```
/* Este ejemplo intenta recopilar lo aprendido de esquemas anteriores
* y comunicará dos NodeMCU station "hablando" entre ellos.
* La acción en uno, establecerá una reacción en el otro.
*/

#include <ESP8266WiFi.h>
extern "C" {
#include <espnow.h>
}

//***ESTRUCTURA DE LOS DATOS TRANSMITIDOS ENTRE LAS UNIDADES***//
// Se establecerá IGUAL en todas
struct ESTRUCTURA_DATOS {
int estado; // 0 --> apagado , 1 --> encendido
};


// variables de elementos hardware
int entrada = D0; // pulsador
int salida = D1; // LED


// *****
// SETUP
```

```
// *****

void setup() {

//***INICIALIZACIÓN DEL PUERTO SERIE***//
Serial.begin(115200); Serial.println();Serial.println();

//*** pinMode ***
pinMode(entrada,INPUT);
pinMode(salida,OUTPUT);

//***INICIALIZACIÓN DEL PROTOCOLO ESP-NOW***//
if (esp_now_init()!=0) {
Serial.println("*** ESP_Now init failed");
ESP.restart();
delay(1);
}

//***DATOS DE LAS MAC (Access Point y Station) del ESP***//
Serial.print("Access Point MAC de este ESP: "); Serial.println(WiFi.softAPmacAddress());
Serial.print("Station MAC de este ESP: "); Serial.println(WiFi.macAddress());

//***DECLARACIÓN DEL PAPEL DEL DISPOSITIVO ESP EN LA COMUNICACIÓN***//
//0=OCIOSO, 1=MAESTRO, 2=ESCLAVO y 3=MAESTRO+ESCLAVO
esp_now_set_self_role(3); // Sería como maestro y esclavo a la vez

//***EMPAREJAMIENTO CON EL ESCLAVO***//
// Dirección MAC del ESP con el que se empareja (esclavo)
// Se debe introducir la STA MAC correspondiente
uint8_t mac_addr[6] = {0xaa, 0xbb, 0xcc, 0xdd, 0xee, 0xff}; // STA MAC esclavo
uint8_t role=2;
uint8_t channel=3;
uint8_t key[0]={}; //no hay clave
//uint8_t key[16] = {0,255,1,1,1,1,1,1,1,1,1,1,1,1,1,1};
uint8_t key_len=sizeof(key);
Serial.print("Tamaño de *key: "); Serial.println(key_len);
esp_now_add_peer(mac_addr,role,channel,key,key_len);
}
```

```
// *******
// LOOP
// *******
void loop() {


// ---------------------------------------------------------------------------
// ENVÍO: como Master
// ---------------------------------------------------------------------------
//***DATOS A ENVIAR***//
ESTRUCTURA_DATOS ED;
ED.estado = digitalRead(entrada);
Serial.print("Dato pulsado: "); Serial.print(ED.estado);
delay(20);

//***ENVÍO DE LOS DATOS***//
//uint8_t *da=NULL; //NULL envía los datos a todos los ESP con los que está emparejado
uint8_t mac_addr[6] = {0xaa, 0xbb, 0xcc, 0xdd, 0xee, 0xff}; // STA MAC esclavo
uint8_t data[sizeof(ED)]; memcpy(data, &ED, sizeof(ED));
uint8_t len = sizeof(data);
esp_now_send(da, data, len);

delay(1); //Si se pierden datos en la recepción se debe subir este valor

// *** CALLBACK de datos ENVIADOS ***
// *** Verificamos que los datos se han enviado....
esp_now_register_send_cb([](uint8_t* mac, uint8_t status) {
char MACesclavo[6];
sprintf(MACesclavo,"%02X:%02X:%02X:%02X:%02X:%02X",mac[0],mac[1],mac[2],mac[3],mac[4],mac[5]);
Serial.print(". Enviado a ESP MAC: "); Serial.print(MACesclavo);
Serial.print(". Recepcion (0=0K - 1=ERROR): "); Serial.println(status);
});
// ---------------------------------------------------------------------------

// ---------------------------------------------------------------------------
// Recepción: como Esclavo
// ---------------------------------------------------------------------------
```

```
esp_now_register_recv_cb([](uint8_t *mac, uint8_t *data, uint8_t len) {
char MACmaestro[6];
sprintf(MACmaestro,
"%02X:%02X:%02X:%02X:%02X:%02X",mac[0],mac[1],mac[2],mac[3],mac[4],mac[5]);
Serial.print("Recepcion desde ESP MAC: "); Serial.print(MACmaestro);

ESTRUCTURA_DATOS ED2;
memcpy(&ED2, data, sizeof(ED2));

Serial.print(". Dato estado: "); Serial.print(ED2.estado);

digitalWrite(salida,ED2.estado);
});
// ---------------------------------------------------------------------------

}
```

view raw NodeMCU_AP+STATION.ino hosted with ❤ by GitHub

El programa del segundo NodeMCU

Es idéntico al primero. Lo único que cambia es que ahora el primer NodeMCU es la estación (STA) vista desde el segundo (AP), así que hay que poner la STA MAC del primero en esp_now_add_peer y esp_now_send.

= = = = = = = = = = = =

NOTA: en estos días de confinamiento forzoso, con la amenaza de una enfermedad infecciosa en las puertas, este resultado me ha puesto muy contento. ¡¡Por fin lo he entendido!!