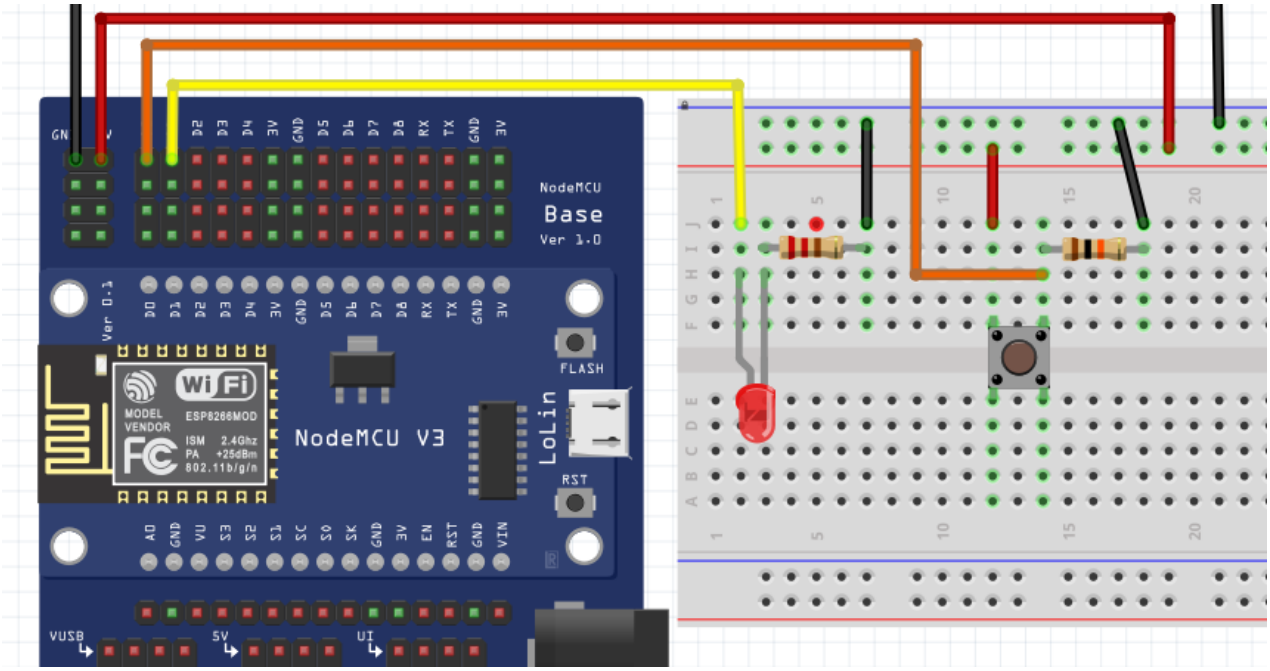
NodeMCU: 11_ESPNOW: Botón del pánico: Central_y_estación. Versión 01

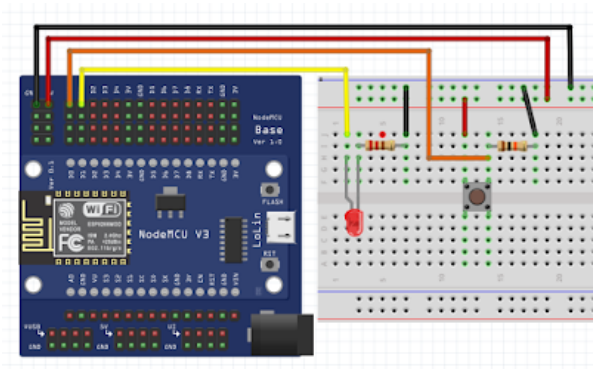
agrportfolioeducativo.blogspot.com/2020/04/nodemcu-11boton-del-panico.html



Introducción

Primera versión del proyecto (V01) terminada. Se hacen ciertas mejoras a ambas partes, CENTRAL y ESTACIÓN, sobre todo a la primera. Es el primer desarrollo funcional. En próximas versiones se podría ampliar a varias estaciones y una estación central.

Estación



```
/* 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 --> operativo, 2 -->activado
String estacion;
};

// variables de elementos hardware
int entrada = D0; // pulsador
int salida = D1; // LED
int pulsado = 0;
int pulsadoAntes = 0;
long tiempo = 0; // variable tiempo
int cadencia=500; // Cadencia de parpadeo
// 500 -> medio segundo
int desbloqueo=0; // veces que es necesario pulsar
int n = 1; // períodos de apagados (tren de pulsos)
// el botón para desbloquear la alarma

ESTRUCTURA_DATOS ED;
ESTRUCTURA_DATOS ED2; // recepcion

// *********
// funciones
// *********


// *****
// 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: CENTRAL***//
// Dirección MAC del ESP con el que se empareja (esclavo)
// Se debe introducir la STA MAC correspondiente
uint8_t mac_addr[6] = {0xEC, 0xFA, 0xBC, 0xC5, 0xAC, 0xAF}; // STA MAC esclavo
// Esta es la MAC de la CENTRAL
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);

ED.estado = 1; // Si termina el SETUP está operativo
ED.estacion="A";

}

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


// ---------------------------------------------------------------------------
// Acciones del programa que no suponen envío de datos
// ---------------------------------------------------------------------------
tiempo = millis();
pulsado = digitalRead(entrada);

// A) caso que ED.estado=1 (operativo y se pulse)
if (pulsado==1 & pulsadoAntes==0 & ED.estado==1) {
```

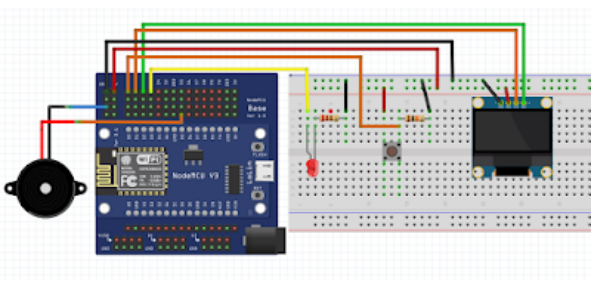
```
ED.estado = 2; // Activo la alarma
} else if (pulsado==1 & pulsadoAntes==0 & ED.estado==2 & desbloqueo<=2) {
// O en caso que pulse el estado sea 2 y no haya desbloqup
desbloqueo+=1; // Suma 1 a desbloqueo
}

// B) Llega al tercer estado de desbloqueo
if (desbloqueo==3) { //
desbloqueo=0;
ED.estado=1; // inactiva la alarma
}

// C) Según el caso cambia el parpadeo
// cadencia = 19900-9900*ED.estado;
n=-8*ED.estado+18; // Si estado 1, n=10; si 2, n=2
// Estado 1 --> Cada 10 segundos; Estado 2 --> cada 0.1 seg
digitalWrite(D1,((ED.estado*tiempo/cadencia)%n==0)); // Un parpadeo cada n veces
Serial.print("Alarma: (1-OFF,2-ON): " + (String) ED.estado + " // desbloqueo: "+ (String) desbloqueo);
delay(10); // pequeño delay...

// ---------------------------------------------------------------------------
// ENVÍO: como Master
// ---------------------------------------------------------------------------

//***ENVÍO DE LOS DATOS***//
//uint8_t *da=NULL; //NULL envía los datos a todos los ESP con los que está emparejado
uint8_t da[6] = {0xEC, 0xFA, 0xBC, 0xC5, 0xAC, 0xAF}; // ¿mismos datos que STA MAC?
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);
memcpy(&ED2, data, sizeof(ED2));
Serial.print(" # Recibido de "+ ED2.estacion + " // Estoy --> "+ (String) ED2.estado);
// digitalWrite(salida,ED2.estado);

});
// ---------------------------------------------------------------------------

// recepción de un 1
// -----------------------------------
if (ED2.estacion=="CENTRAL" and ED2.estado==1) {
// apaga la alarma
desbloqueo=0;
ED.estado=1; // inactiva la alarma
}

// -----------------------------------

pulsadoAntes=pulsado; // Importante para reconocer el flanco de subida del pulsador

Serial.println();
}
```

view raw NodeMCU_BTNPA_EMISOR_V01.ino hosted with ❤ by GitHub

En el caso de la estación, el programa básico no ha cambiado. Excepto que la estructura de datos en la recepción se ha cambiado y puesto como variable global y la rutina que apaga la alarma controlada desde la CENTRAL.

Central



```
/* 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.
*
* Por Aurelio Gallardo, 19-Abril-2020
*/

/* Biblioteca para la comunicación Wifi */
#include <ESP8266WiFi.h>
extern "C" {
#include <espnow.h>
}

/* Bibliotecas para comunicarse con el módulo OLED */
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h> //Nucleo de la librería gráfica.
#include <Adafruit_SSD1306.h> //Librería para pantallas OLED monocromas de 128x64 y 128x32
// #include <Fonts/FreeMonoBoldOblique12pt7b.h> // Anuladas0
#include <Fonts/FreeSerifBold9pt7b.h> // Biblioteca de fuentes: https://learn.adafruit.com/adafruit-gfx-graphics-library/using-fonts

//
================================================================================================
// IMPORTANTE: definición de pantalla, y RESET que funciona, sin tocar librería adafruit_SSD1306
//
================================================================================================
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels

// Declaration for an SSD1306 display connected to I2C (SDA, SCL pins)
#define OLED_RESET 0 // Reset pin # (or -1 if sharing Arduino reset pin)
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
//
================================================================================================

#define NUMFLAKES 10
#define XPOS 0
#define YPOS 1
#define DELTAY 2

const unsigned char PROGMEM logo [] = {
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x06, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x3F, 0xC0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
```

0x00, 0x00, 0x00, 0x00, 0x00, 0x3F, 0xF0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x7F, 0xFC, 0x00, 0x00, 0x38, 0x00, 0x10, 0x00, 0x20, 0x40, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFC, 0x00, 0x00, 0x6C, 0x06, 0x20, 0x00, 0x20, 0xC0, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFE, 0x00, 0x00, 0x6C, 0xEF, 0x77, 0x81, 0xEE, 0xC0, 0x00,

0x00, 0x00, 0x00, 0x7E, 0x00, 0xFF, 0xFE, 0x00, 0x00, 0x7D, 0xA6, 0xD6, 0x83, 0x7E, 0xC0, 0x00,

0x00, 0x00, 0x00, 0xFF, 0x01, 0xFF, 0xFE, 0x00, 0x00, 0x65, 0xB6, 0xDE, 0x82, 0x7C, 0xC0, 0x00,

0x00, 0x00, 0x03, 0xFF, 0xC0, 0xFF, 0xFE, 0x00, 0x00, 0x6D, 0xA6, 0xD6, 0xC3, 0x78, 0xC0, 0x00,

0x00, 0x00, 0x03, 0xFF, 0xF0, 0xFF, 0xFF, 0x00, 0x00, 0x38, 0xE2, 0x76, 0xC1, 0xEF, 0x40, 0x00,

0x00, 0x00, 0x03, 0xFF, 0xF0, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0x70, 0x03, 0xFF, 0xF0, 0x7F, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x03, 0xFC, 0x07, 0xFF, 0xF8, 0x3F, 0xFF, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x03, 0xFE, 0x07, 0xFF, 0xF8, 0x1F, 0xFF, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x0F, 0xFF, 0x07, 0xFF, 0xF8, 0x07, 0xFF, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x0F, 0xFF, 0xC7, 0xFF, 0xFC, 0xE3, 0xFF, 0xC0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x1F, 0xFF, 0xC3, 0xFF, 0xFC, 0xE0, 0x7F, 0xC0, 0x00, 0x00, 0x00, 0x80, 0x80, 0x00, 0x00, 0x00,

0x1F, 0xFF, 0xC1, 0xFF, 0xFC, 0xFC, 0x1F, 0xC0, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00,

0x1F, 0xFF, 0xE0, 0xFF, 0xFE, 0x7F, 0x03, 0xE0, 0x00, 0x00, 0xF3, 0xBC, 0x9C, 0xE0, 0x00, 0x00,

0x1F, 0xFF, 0xE0, 0x7F, 0xFE, 0x7F, 0xC0, 0xE0, 0x00, 0x00, 0xDE, 0xF4, 0xB1, 0xA0, 0x00, 0x00,

0x0F, 0xFF, 0xE0, 0x1F, 0xFE, 0x7F, 0xF8, 0x40, 0x00, 0x00, 0xDE, 0xB5, 0xB1, 0xB0, 0x00, 0x00,

0x0F, 0xFF, 0xF7, 0x8F, 0xFF, 0x3F, 0xFC, 0x00, 0x00, 0x00, 0xDE, 0xF7, 0xB1, 0xA0, 0x00, 0x00,

0x0F, 0xFF, 0xF7, 0xC0, 0xFF, 0x3F, 0xFF, 0x80, 0x00, 0x00, 0xF3, 0xF6, 0x9C, 0xE0, 0x00, 0x00,

0x07, 0xFF, 0xF7, 0xF8, 0x1F, 0x3F, 0xFF, 0xC0, 0x00, 0x00, 0xC0, 0x00, 0x00, 0x00, 0x00, 0x00,

0x03, 0xFF, 0xFB, 0xFE, 0x0F, 0x1F, 0xFF, 0xE0, 0x00, 0x00, 0xC0, 0x00, 0x00, 0x00, 0x00, 0x00,

0x00, 0xFF, 0xFB, 0xFF, 0x83, 0x9F, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x08, 0xFF, 0xFB, 0xFF, 0xE0, 0x1F, 0xFF, 0xF8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

0x0E, 0x0F, 0xFD, 0xFF, 0xF8, 0x0F, 0xFF, 0xF8, 0x00, 0x00, 0x00, 0x1E, 0x00, 0x00, 0x00, 0x00,

0x0F, 0x03, 0xFD, 0xFF, 0xFF, 0x0F, 0xFF, 0xFC, 0x00, 0x00, 0x00, 0x3F, 0x00, 0x00, 0x00, 0x00,

0x0F, 0xF0, 0xFD, 0xFF, 0xFF, 0x0F, 0xFF, 0xFC, 0x00, 0x00, 0x00, 0x3F, 0x80, 0x00, 0x00, 0x00,

0x07, 0xF8, 0x1C, 0xFF, 0xFF, 0x8F, 0xFF, 0xFC, 0x00, 0x00, 0x00, 0x3F, 0x80, 0x00, 0x00, 0x00,

0x07, 0xFF, 0x04, 0xFF, 0xFF, 0xC7, 0xFF, 0xF8, 0x00, 0x00, 0x00, 0x3F, 0x80, 0x00, 0x00, 0x00,

0x07, 0xFF, 0xE0, 0xFF, 0xFF, 0xE7, 0xFF, 0xF0, 0x00, 0x00, 0x00, 0x3F, 0x00, 0x00, 0x00, 0x00,

0x03, 0xFF, 0xF0, 0xFF, 0xFF, 0xE7, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x1E, 0x00, 0x00, 0x00, 0x00,

0x03, 0xFF, 0xFE, 0x7F, 0xFF, 0xF3, 0xF8, 0x00, 0x00, 0x00, 0x00, 0x07, 0xC0, 0x00, 0x00, 0x00,

0x03, 0xFF, 0xFE, 0x7F, 0xFF, 0xF3, 0xC0, 0x00, 0x00, 0x00, 0x00, 0x07, 0xF8, 0x00, 0x00, 0x00,

0x01, 0xFF, 0xFE, 0x7F, 0xFF, 0xE0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0xFC, 0x00, 0x00, 0x00,

0x01, 0xFF, 0xFF, 0x3F, 0xFF, 0xE0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0xFC, 0x00, 0x00, 0x00,

0x01, 0xFF, 0xFF, 0x3F, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x37, 0xFC, 0x00, 0x00, 0x00,

0x01, 0xFF, 0xFF, 0x3F, 0xF8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3F, 0xFC, 0x00, 0x00, 0x00,

0x00, 0xFF, 0xFF, 0x9F, 0xC0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1F, 0xFC, 0x00, 0x00, 0x00,

```
0x00, 0xFF, 0xFF, 0x9C, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0F, 0xFC, 0x00, 0x00, 0x00,
0x00, 0xFF, 0xFF, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0xF8, 0x00, 0x00, 0x00,
0x00, 0x7F, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0xF8, 0x00, 0x00, 0x00,
0x00, 0x7F, 0xF8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xF8, 0x00, 0x00, 0x00,
0x00, 0x7F, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xF8, 0x00, 0x00, 0x00,
0x00, 0x3C, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x20, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x70, 0x00, 0x20, 0x00, 0x0A, 0x00, 0x38, 0x0A, 0x00, 0x08, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x48, 0x00, 0x60, 0x00, 0x08, 0x00, 0x68, 0x0A, 0x00, 0x08, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x5A, 0x20, 0xE2, 0x9D, 0xCA, 0x70, 0x41, 0xCA, 0xE7, 0x7B, 0x80,
0x00, 0x00, 0x00, 0x00, 0x00, 0x72, 0x40, 0xA4, 0xB5, 0x4A, 0xD0, 0xFF, 0x4B, 0xAD, 0x5E, 0x80,
0x00, 0x00, 0x00, 0x00, 0x00, 0x49, 0x41, 0xF4, 0xB3, 0x8A, 0x90, 0x8E, 0x4B, 0x2C, 0x8C, 0x80,
0x00, 0x00, 0x00, 0x00, 0x00, 0x49, 0x81, 0x12, 0xB3, 0x2A, 0xD0, 0xCA, 0x4B, 0x2C, 0xDE, 0x80,
0x00, 0x00, 0x00, 0x00, 0x00, 0x70, 0x81, 0x13, 0x91, 0xCA, 0x70, 0x71, 0xCA, 0xE4, 0x7B, 0x80,
0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00
};

//***ESTRUCTURA DE LOS DATOS TRANSMITIDOS ENTRE LAS UNIDADES***//
// Se establecerá IGUAL en todas
struct ESTRUCTURA_DATOS {
int estado; // 1 --> operativo, 2 --> activo
String estacion;
};


// variables de elementos hardware
int entrada = D0; // pulsador
int salida = D3; // LED
int pulsado = 0;
int pulsadoAntes = 0;

// variables de sirena
unsigned int frecmin = 2000;
unsigned int frecmax = 4000;
unsigned int duracion = 250;
```

```
ESTRUCTURA_DATOS ED; // Para el envío
ESTRUCTURA_DATOS ED2; // Para la recepción

// variables temporales
unsigned long tiempo = 0;
int x = 0;
int y = 0;

// *****
// 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] = {0xA4, 0xCF, 0x12, 0xDF, 0x5A, 0x6B}; // 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);
ED.estacion="CENTRAL";

/*** Inicialización de pantalla OLED ***/
// Inicializa pantalla con en la dirección 0x3C para la conexión I2C.
display.begin(SSD1306_SWITCHCAPVCC, 0x3C);

// Visualización de un LOGO
presentarLogo();
// testdrawchar();
delay(5000); // Tiempo para ver el logotipo
}

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

tiempo = millis();

pulsado = digitalRead(entrada);

// ---------------------------------------------------------------------------
// ENVÍO: como Master
// ---------------------------------------------------------------------------
//***DATOS A ENVIAR***//
if (pulsado==1 & pulsadoAntes==0) {
ED.estado = 1-ED.estado;
// Serial.print("Dato pulsado: "); Serial.println(ED.estado);
// Serial.print("Variable pulsado: "); Serial.println(pulsado);
// Serial.print("Variable pulsadoAntes: "); Serial.println(pulsadoAntes);
}
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 da[6] = {0xA4, 0xCF, 0x12, 0xDF, 0x5A, 0x6B}; // ¿mismos datos que STA MAC?
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);
memcpy(&ED2, data, sizeof(ED2));
// Serial.println("Recibiendo de " + ED2.estacion + " // Su estado: "+ (String) ED2.estado);
digitalWrite(salida,ED2.estado-1); // -1 porque recibe un dos cuando está ACTIVADO
});
// -----------------------------------------------------------------------------

// -----------------------------------------------------------------------------
/*** DIBUJANDO EN PANTALLA ***/
if ((((tiempo/1000))%6)==0) { // una vez cada 6 segundos presenta la pantalla
presentarLogo();
} else {
presentarPantalla(ED2.estacion, (String) ED2.estado);
}
if (ED2.estado==2) {
int signo = (tiempo%(frecmax-frecmin)==0)*(1-signo);
unsigned int valorTono = (frecmin+(tiempo%(frecmax-frecmin)))*(signo==0)+(frecmax-((frecmax-frecmin)%1000))*
(signo==1);
Serial.println(valorTono);
tone(D5,valorTono,duracion);
} else {noTone(D5);}
```

```
// ---------------------------------------------------------------------------
// ---------------------------------------------------------------------------
// Si la recepción es el estado apagado, fuerzo al estado del botón a cero.
if (ED2.estado==1) {ED.estado=0;}
// ---------------------------------------------------------------------------
/*** ULTIMA LINEA ***/
pulsadoAntes=pulsado;
}

/* ---------------------------------------------------------------------------*/
/*** Funciones ***/
/* ---------------------------------------------------------------------------*/
void presentarPantalla(String emisor, String estado) {
String cadena[]={"Desc.","OFF","ON"};
display.fillScreen(0); //Limpiamos la pantalla
// display.setFont(&FreeMonoBoldOblique12pt7b); // Quito las fuentes de pantalla
// https://randomnerdtutorials.com/esp8266-0-96-inch-oled-display-with-arduino-ide/
display.cp437(true); // Use full 256 char 'Code Page 437' font
display.setTextSize(2); // Tamaño
display.setTextColor(1,0); // Color (1,0) ó 0,1 invertido
display.setCursor(x,y);
display.print(F("Estaci"));
display.write(162);
display.print(F("n "));
display.println(ED2.estacion);
display.setCursor(x,y+30);
display.println("Estado " + cadena[ED2.estado]);
display.display();
}

void presentarLogo(){
// Visualización de un LOGO
display.fillScreen(0); //Limpiamos la pantalla
display.drawBitmap(0,0,logo,128,64,1); // drawBitmap (int16_t x, int16_t y, uint8_t * mapa de bits, int16_t w, int16_t h,
uint16_t color)
// https://www.brainy-bits.com/create-arduino-array-from-pictures/
display.display(); // Inicializa
}


void testdrawchar(void) {
```

```
    display.setTextSize(2); // Normal 1:1 pixel scale
    display.setTextColor(WHITE); // Draw white text
    // Start at top-left corner
    display.cp437(true); // Use full 256 char 'Code Page 437' font

    // Not all the characters will fit on the display. This is normal.
    // Library will draw what it can and the rest will be clipped.
    for(int16_t i=0; i<256; i++) {
    display.clearDisplay();
    display.setCursor(0, 0);
    display.print((String) i);
    if(i == '\n') display.write(' ');
    else display.write(i);
    delay(1000);
    display.display();
    }

    }
```

view raw NodeMCU_BTNPA_CENTRAL_CON_OLED__V01.ino hosted with ❤ by GitHub

En este caso, el programa sufre varias ampliaciones

1. Se dibuja un logo nuevo para el programa.
2. Se modifica la salida del LED (pasa a D3) y en D5 ponemos un buzzer.
3. Creamos tres variables: frecmin, frecmax y duracion, para poder modificar el sonido de la sirena. Dejo los valores en 2000, 4000 y 250, pero se puede jugar con ellos. 700 y 800 dan un sonido más de "cacharros de feria".
4. La estructura de datos E2 pasa a ser variable global.
5. Hay una rutina por la que cada seis segundos muestra el logo (función presentarLogo). Así sabremos que no se ha bloqueado.
6. En otra rutina (presentarPantalla) presentamos en la pantalla la información de la estación. Se ha mejorado incluyendo el símbolo ASCII de la "ó".
7. No he borrado la función de testeo. Hay memoria para ello y me parece útil si quiero buscar un símbolo (testdrawchar)
8. En la línea 243, si tenemos estado de alarma, suena el buzzer.
9. En la línea 252, si el estado que recibo es de apagado, fuerzo a que la alarma se desactive.

<