Nodo_Luz

Andrés Felipe Fernández Ríos
Elkin Alejandro Ledesma Narvaez
Angie Tatiana Perez Muñoz

Versión 1.0.0

Índice de clases

Lista de clases

Lista de clases, estructuras, uniones e interfaces con breves descripciones:

Índice de archivos

Lista de archivos

Lista de todos los archivos con breves descripciones:

Documentación de clases

Referencia de la estructura light_message_t

Estructura de datos a enviar vía ESP-NOW.

Atributos públicos

- int **ldrValue**

Descripción detallada

Estructura de datos a enviar vía ESP-NOW.

Contiene un único valor entero correspondiente a la lectura del LDR.

Definición en la línea **28** del archivo **Nodo_Luz.ino**.

Documentación de datos miembro

int light_message_t::ldrValue

Valor de luminosidad leído del sensor

Definición en la línea **29** del archivo **Nodo_Luz.ino**.

La documentación de esta estructura está generada del siguiente archivo:

- C:/Users/Andres Fernandez/Documents/Arduino/sketch_may7a/Nodo_Luz/**Nodo_Luz.ino**

Documentación de archivos

Referencia del archivo C:/Users/Andres Fernandez/Documents/Arduino/sketch_may7a/Nodo_Luz/Nodo_Luz.ino

Nodo emisor ESP32 que mide luminosidad con un LDR y envía los datos al receptor mediante ESP-NOW.

```
#include <esp_now.h>
#include <WiFi.h>
#include "esp_wifi.h"
```

Clases

struct **light_message_t***Estructura de datos a enviar vía ESP-NOW.*

defines

- #define **LDR_PIN** 32
  Pin analógico conectado al LDR.

Funciones

- void **OnDataSent** (const uint8_t *mac_addr, esp_now_send_status_t status)
  Callback llamado tras enviar un paquete ESP-NOW.

- void **LDRTask** (void *parameter)
  Tarea que lee el valor del LDR y lo envía periódicamente.

- void **setup** ()
  Función de configuración inicial.

- void **loop** ()
  Bucle principal (no se usa en este diseño basado en tareas).

Variables

- uint8_t **receiverMAC** [] = {0x5C, 0x01, 0x3B, 0x72, 0xF2, 0xCC}
  Dirección MAC del nodo receptor al que se envían los datos.

Descripción detallada

Nodo emisor ESP32 que mide luminosidad con un LDR y envía los datos al receptor mediante ESP-NOW.

Este sketch configura un pin analógico para leer un sensor de luminosidad (LDR), envía la medición periódicamente al nodo receptor usando ESP-NOW y reporta el estado por Serial.

Definición en el archivo **Nodo_Luz.ino**.

Documentación de «define»

#define LDR_PIN 32

Pin analógico conectado al LDR.

Definición en la línea **14** del archivo **Nodo_Luz.ino**.

Documentación de funciones

void LDRTask (void * parameter)

Tarea que lee el valor del LDR y lo envía periódicamente.

Lee el pin analógico, asigna el valor a la estructura sensorSend y envía el dato al nodo receptor mediante esp_now_send. La tarea se repite cada 10 segundos.

Parámetros

parameter	Parámetro de tarea no usado

Definición en la línea **58** del archivo **Nodo_Luz.ino**.

void loop ()

Bucle principal (no se usa en este diseño basado en tareas).

Definición en la línea **138** del archivo **Nodo_Luz.ino**.

void OnDataSent (const uint8_t * mac_addr, esp_now_send_status_t status)

Callback llamado tras enviar un paquete ESP-NOW.

Informa por Serial si el envío fue exitoso o falló.

Parámetros

mac_addr	Dirección MAC del peer destino
status	Estado del envío (ESP_NOW_SEND_SUCCESS o fallo)

Definición en la línea **43** del archivo **Nodo_Luz.ino**.

void setup ()

Función de configuración inicial.

Configura Serial, inicializa WiFi en modo STA, fuerza el canal WiFi, inicia ESP-NOW, registra el callback y agrega el peer receptor. Crea la tarea LDRTask en el Core 1.

< Función de la tarea

< Nombre de la tarea

< Tamaño de stack

< Parámetro

< Prioridad

< Handle

< Núcleo

Definición en la línea **87** del archivo **Nodo_Luz.ino**.

Documentación de variables

uint8_t receiverMAC[] = {0x5C, 0x01, 0x3B, 0x72, 0xF2, 0xCC}

Dirección MAC del nodo receptor al que se envían los datos.

Definición en la línea **20** del archivo **Nodo_Luz.ino**.

Nodo_Luz.ino

Ir a la documentación de este archivo.

```
00001
00008
00009 #include <esp_now.h>
00010 #include <WiFi.h>
00011 #include "esp_wifi.h"
00012
00014 #define LDR_PIN 32
00015
00017 static const uint8_t WIFI_CHANNEL = 6;
00018
00020 uint8_t receiverMAC[] = {0x5C, 0x01, 0x3B, 0x72, 0xF2, 0xCC};
00021
00028 typedef struct {
00029   int ldrValue;
00030 } light_message_t;
00031
00033 static light_message_t sensorSend;
00034
00043 void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
00044   Serial.printf("[ESP-NOW] Envío a %02X:%02X:%02X:%02X:%02X:%02X -> %s\n",
00045                 mac_addr[0], mac_addr[1], mac_addr[2],
00046                 mac_addr[3], mac_addr[4], mac_addr[5],
00047                 status == ESP_NOW_SEND_SUCCESS ? "OK" : "FALLÓ");
00048 }
00049
00058 void LDRTask(void *parameter) {
00059   (void) parameter;  // Evitar advertencia de parámetro no usado
00060
00061   for (;;) {
00062     // Leer valor analógico del LDR
00063     int ldr = analogRead(LDR_PIN);
00064     sensorSend.ldrValue = ldr;
00065
00066     // Mostrar valor en el monitor Serial
00067     Serial.printf("[LDR] Valor: %d\n", ldr);
00068
00069     // Enviar estructura al receptor
00070     esp_err_t result = esp_now_send(receiverMAC, (uint8_t *)&sensorSend,
sizeof(sensorSend));
00071     if (result != ESP_OK) {
00072       Serial.printf("[Error] Fallo al enviar datos (err %d)\n", result);
00073     }
00074
00075     // Esperar 10 segundos antes de la siguiente lectura
00076     vTaskDelay(pdMS_TO_TICKS(10000));
00077   }
00078 }
00079
00087 void setup() {
00088   // Inicializar comunicación Serial
00089   Serial.begin(115200);
00090   delay(100);
00091
00092   // Configurar pin LDR
00093   pinMode(LDR_PIN, INPUT);
00094
00095   // Modo estación WiFi y mostrar MAC
00096   WiFi.mode(WIFI_STA);
00097   Serial.printf("[Setup] MAC STA: %s\n", WiFi.macAddress().c_str());
00098
00099   // Forzar canal WiFi para ESP-NOW
00100   esp_wifi_set_channel(WIFI_CHANNEL, WIFI_SECOND_CHAN_NONE);
00101   Serial.printf("[Setup] Canal WiFi: %d\n", WIFI_CHANNEL);
00102
00103   // Inicializar ESP-NOW
00104   if (esp_now_init() != ESP_OK) {
00105     Serial.println("[Error] No se pudo iniciar ESP-NOW");
00106     return;
00107   }
00108
```

```
00109   // Registrar callback de envío
00110   esp_now_register_send_cb(OnDataSent);
00111
00112   // Configurar y añadir peer receptor
00113   esp_now_peer_info_t peerInfo = {};
00114   memcpy(peerInfo.peer_addr, receiverMAC, 6);
00115   peerInfo.channel = WIFI_CHANNEL;
00116   peerInfo.encrypt = false;
00117
00118   if (esp_now_add_peer(&peerInfo) != ESP_OK) {
00119     Serial.println("[Error] Fallo al agregar peer receptor");
00120     return;
00121   }
00122
00123   // Crear tarea LDRTask en Core 1
00124   xTaskCreatePinnedToCore(
00125     LDRTask,
00126     "LDRTask",
00127     2048,
00128     NULL,
00129     1,
00130     NULL,
00131     1
00132   );
00133 }
00134
00138 void loop() {
00139   // No hacer nada, la tarea LDRTask se encarga de todo
00140   vTaskDelay(portMAX_DELAY);
00141 }
```