MCGA

NODO DTH+LDR+MQ5

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

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 struct_message_send

Estructura que agrupa las mediciones de sensores para envío.

Atributos públicos

- float **temp**
- float **hum**
- int **luz**
- int **mq5**

Descripción detallada

Estructura que agrupa las mediciones de sensores para envío.

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

Documentación de datos miembro

float struct_message_send::hum

Humedad relativa en %.

Definición en la línea **45** del archivo **NodoDTH.ino**.

int struct_message_send::luz

Valor analógico leído del LDR (0-4095).

Definición en la línea **46** del archivo **NodoDTH.ino**.

int struct_message_send::mq5

Valor analógico leído del MQ5 (0-4095).

Definición en la línea **47** del archivo **NodoDTH.ino**.

float struct_message_send::temp

Temperatura ambiental en °C.

Definición en la línea **44** del archivo **NodoDTH.ino**.

La documentación de esta estructura está generada del siguiente archivo:

- C:/Users/Andres Fernandez/Desktop/ProyectoMGA/NodoDTH/**NodoDTH.ino**

Documentación de archivos

Referencia del archivo C:/Users/Andres Fernandez/Desktop/ProyectoMGA/NodoDTH/NodoDTH.ino

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

Clases

struct **struct_message_send***Estructura que agrupa las mediciones de sensores para envío.*

defines

- #define **DHTPIN** 5
  Pin digital conectado al sensor DHT (GPIO).

- #define **DHTTYPE** DHT22
  Tipo de sensor DHT (DHT11 o DHT22).

- #define **MQ5_PIN** 34
  Pin analógico conectado al sensor MQ-5 (gas/metano).

- #define **LDR_PIN** 32
  Pin analógico conectado al sensor LDR (luminosidad).

Funciones

- DHT **dht** (**DHTPIN**, **DHTTYPE**)
  Objeto DHT para lectura de temperatura y humedad.

- void **OnDataSent** (const uint8_t *mac_addr, esp_now_send_status_t status)
  Callback invocado tras intentar enviar datos por ESP-NOW.

- void **setup** ()
  Función setup de Arduino.

- void **loop** ()
  Función loop de Arduino.

Variables

- uint8_t **receiverMAC** [] = {0x5c, 0x01, 0x3b, 0x72, 0xf2, 0xcc}
  Dirección MAC del ESP32 receptor para ESP-NOW.

- **struct_message_send sensorSend**
  Variable global que almacena la estructura de datos a enviar.

Documentación de «define»

#define DHTPIN 5

Pin digital conectado al sensor DHT (GPIO).
Definición en la línea **19** del archivo **NodoDTH.ino**.

#define DHTTYPE DHT22

Tipo de sensor DHT (DHT11 o DHT22).
Definición en la línea **21** del archivo **NodoDTH.ino**.

#define LDR_PIN 32

Pin analógico conectado al sensor LDR (luminosidad).
Definición en la línea **33** del archivo **NodoDTH.ino**.

#define MQ5_PIN 34

Pin analógico conectado al sensor MQ-5 (gas/metano).
Definición en la línea **31** del archivo **NodoDTH.ino**.

Documentación de funciones

DHT dht (DHTPIN , DHTTYPE)

Objeto DHT para lectura de temperatura y humedad.
Utiliza el pin y tipo definido en DHTPIN y DHTTYPE.

void loop ()

Función loop de Arduino.

Nota

No se utiliza porque se entra en deep sleep y reset cada ciclo.

Definición en la línea **143** del archivo **NodoDTH.ino**.

void OnDataSent (const uint8_t * mac_addr, esp_now_send_status_t status)

Callback invocado tras intentar enviar datos por ESP-NOW.

Parámetros

mac_addr	Dirección MAC del receptor.
status	Estado del envío (éxito o fallo).

Muestra en el monitor serial si el paquete se envió correctamente.

Definición en la línea **61** del archivo **NodoDTH.ino**.

void setup ()

Función setup de Arduino.

Inicializa Serial, DHT, WiFi en modo STA, ESP-NOW, configura peer, lee sensores, envía datos y programa deep sleep.

< Tiempo para estabilizar serial.

< Canal ESP-NOW fijo para estabilidad.

< Lectura temperatura (°C).

< Lectura humedad (%).

< Lectura LDR.

< Lectura MQ-5.

< Deep sleep: 30 segundos en microsegundos.

Definición en la línea **73** del archivo **NodoDTH.ino**.

Documentación de variables

uint8_t receiverMAC[] = {0x5c, 0x01, 0x3b, 0x72, 0xf2, 0xcc}

Dirección MAC del ESP32 receptor para ESP-NOW.

Definición en la línea **37** del archivo **NodoDTH.ino**.

struct_message_send sensorSend

Variable global que almacena la estructura de datos a enviar.

Definición en la línea **51** del archivo **NodoDTH.ino**.

NodoDTH.ino

Ir a la documentación de este archivo.

```
00001
00011
00012 #include <esp_now.h>
00013 #include <WiFi.h>
00014 #include <DHT.h>
00015 #include "esp_wifi.h"
00016
00017 // ── Configuración del sensor DHT ──
00019 #define DHTPIN    5
00021 #define DHTTYPE   DHT22
00022
00027 DHT dht(DHTPIN, DHTTYPE);
00028
00029 // ── Configuración de pines analógicos ──
00031 #define MQ5_PIN   34
00033 #define LDR_PIN   32
00034
00035 // ── Dirección MAC del receptor ──
00037 uint8_t receiverMAC[] = {0x5c, 0x01, 0x3b, 0x72, 0xf2, 0xcc};
00038
00039 // ── Estructura de datos a enviar ──
00043 typedef struct {
00044   float temp;
00045   float hum;
00046   int luz;
00047   int mq5;
00048 } struct_message_send;
00049
00051 struct_message_send sensorSend;
00052
00053 // ── Callback de envío ──
00061 void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
00062   Serial.printf("ðŸ“¡ EnvÃo ESP-NOW: %s\n",
00063                 status == ESP_NOW_SEND_SUCCESS ? "OK" : "FALLÓ");
00064 }
00065
00066 // ── Función setup principal ──
00073 void setup() {
00074   // Inicialización Serial
00075   Serial.begin(115200);
00076   delay(100);
00077
00078   // Inicialización del sensor DHT
00079   dht.begin();
00080
00081   // Configuración WiFi y ESP-NOW
00082   WiFi.mode(WIFI_STA);
00083   Serial.print("Transmisor MAC STA: ");
00084   Serial.println(WiFi.macAddress());
00085
00086   const uint8_t canal = 6;
00087   esp_wifi_set_channel(canal, WIFI_SECOND_CHAN_NONE);
00088   Serial.printf("Transmisor foro canal: %d\n", canal);
00089
00090   if (esp_now_init() != ESP_OK) {
00091     Serial.println("â Œ Error al iniciar ESP-NOW");
00092     while (true) { delay(1000); }
00093   }
00094   esp_now_register_send_cb(OnDataSent);
00095
00096   // Añadir receptor como peer
00097   esp_now_peer_info_t peerInfo = {};
00098   memcpy(peerInfo.peer_addr, receiverMAC, 6);
00099   peerInfo.channel = canal;
00100   peerInfo.encrypt = false;
00101   if (esp_now_add_peer(&peerInfo) != ESP_OK) {
00102     Serial.println("â Œ Fallo al agregar peer");
00103     while (true) { delay(1000); }
00104   }
00105
```

```
00106   // ── Lectura de sensores ──
00107   float t = dht.readTemperature();
00108   float h = dht.readHumidity();
00109   int luz_val = analogRead(LDR_PIN);
00110   int mq5_val = analogRead(MQ5_PIN);
00111
00112   if (isnan(t) || isnan(h)) {
00113     Serial.println("â□Œ Error al leer DHT22");
00114   } else {
00115     // Preparar estructura y enviar
00116     sensorSend.temp = t;
00117     sensorSend.hum  = h;
00118     sensorSend.luz  = luz_val;
00119     sensorSend.mq5  = mq5_val;
00120
00121     Serial.printf("Temp: %.1f °C | Hum: %.1f %% | Luz: %d | MQ-5: %d\n",
00122                   sensorSend.temp, sensorSend.hum, sensorSend.luz,
sensorSend.mq5);
00123
00124     esp_err_t res = esp_now_send(receiverMAC,(uint8_t
*)&sensorSend,sizeof(sensorSend));
00125     if (res != ESP_OK) {
00126       Serial.println("â□Œ Error al enviar datos");
00127     }
00128   }
00129
00130   // Dar tiempo a ESP-NOW para completar envío
00131   delay(100);
00132
00133   // Entrar en deep sleep
00134   Serial.println("ðŸ˜´ Entrando en deep sleep por 30 segundos...");
00135   esp_deep_sleep(30 * 1000000ULL);
00136 }
00137
00138 // ── Función loop vacía ──
00143 void loop() {
00144   // No implementado
00145 }
```