Nodo_Edge_Doxy

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

Versión 1.0.0

Documentación de clases

Referencia de la estructura dht_message_t

Estructuras para los datos recibidos desde los nodos.

Atributos públicos

- float **temp**
- float **hum**

Descripción detallada

Estructuras para los datos recibidos desde los nodos.

Definición en la línea **45** del archivo **sketch_may7a.ino**.

Documentación de datos miembro

float dht_message_t::hum

Definición en la línea **45** del archivo **sketch_may7a.ino**.

float dht_message_t::temp

Definición en la línea **45** del archivo **sketch_may7a.ino**.

La documentación de esta estructura está generada del siguiente archivo:

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

Referencia de la estructura light_message_t

Atributos públicos

- int **ldrValue**

Descripción detallada

Definición en la línea **47** del archivo **sketch_may7a.ino**.

Documentación de datos miembro

int light_message_t::ldrValue

Definición en la línea **47** del archivo **sketch_may7a.ino**.

La documentación de esta estructura está generada del siguiente archivo:

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

Referencia de la estructura soil_message_t

Atributos públicos

- float **soil**

Descripción detallada

Definición en la línea **46** del archivo **sketch_may7a.ino**.

Documentación de datos miembro

float soil_message_t::soil

Definición en la línea **46** del archivo **sketch_may7a.ino**.

La documentación de esta estructura está generada del siguiente archivo:

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

Documentación de archivos

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

```
#include <ESP8266WiFi.h>
#include <esp_now.h>
#include <WiFiClientSecure.h>
#include <UniversalTelegramBot.h>
#include <time.h>
#include "esp_wifi.h"
```

Clases

struct **dht_message_t** *Estructuras para los datos recibidos desde los nodos.*

- struct **soil_message_t**struct **light_message_t**

defines

- #define **BOTtoken** "7672713502:AAH55Vyi1rU-EF1eij3dnpqYvRMLxzsE58g"
  Token del bot de Telegram y ID de chat donde se enviarán las notificaciones.

- #define **CHAT_ID** "5365006160"
- #define **TEMP_THRESHOLD** 27.0
  Umbrales predefinidos para generar alertas.

- #define **HUM_THRESHOLD** 80.0
- #define **SOIL_THRESHOLD_LOW** 30.0
- #define **SOIL_THRESHOLD_HIGH** 80.0
- #define **LDR_THRESHOLD** 700

Funciones

- UniversalTelegramBot **bot** (**BOTtoken**, **client**)
- void **OnDataRecv** (const esp_now_recv_info_t *info, const uint8_t *incoming, int len)
  Callback llamado automáticamente por ESP-NOW al recibir datos.

- void **DHTTask** (void *parameter)
  Tarea que procesa los datos del sensor DHT.

- void **SoilTask** (void *parameter)
  Tarea que procesa los datos del sensor de humedad de suelo.

- void **LDRTask** (void *parameter)
  Tarea que procesa los datos del sensor LDR (luz).

- void **TelegramTask** (void *parameter)
  Tarea dedicada al envío de mensajes por Telegram.

- void **setup** ()
  Función principal de inicialización.

- void **loop** ()
  Bucle principal del programa (no hace nada porque todo se gestiona con tareas).

Variables

- const char * **ssid** = "iPhone"
  Libresias a utilizar para cada uno de los componentes.

- const char * **password** = "25310tati"
- WiFiClientSecure **client**
- uint8_t **dhtSenderMAC** [] = {0x08, 0xD1, 0xF9, 0xEE, 0x0A, 0x04}
  MAC addresses de los nodos emisores.

- uint8_t **soilSenderMAC** [] = {0xa0, 0xb7, 0x65, 0x1b, 0x18, 0x50}
- uint8_t **ldrSenderMAC** [] = {0xa0, 0xa3, 0xb3, 0x2a, 0xE0, 0x20}
- QueueHandle_t **dhtQueue**
  Colas para manejar los datos recibidos por cada tipo de sensor.

- QueueHandle_t **soilQueue**
- QueueHandle_t **ldrQueue**
- QueueHandle_t **msgQueue**

Documentación de «define»

#define BOTtoken "7672713502:AAH55Vyi1rU-EF1eij3dnpqYvRMLxzsE58g"

Token del bot de Telegram y ID de chat donde se enviarán las notificaciones.

Definición en la línea **26** del archivo **sketch_may7a.ino**.

#define CHAT_ID "5365006160"

Definición en la línea **27** del archivo **sketch_may7a.ino**.

#define HUM_THRESHOLD 80.0

Definición en la línea **34** del archivo **sketch_may7a.ino**.

#define LDR_THRESHOLD 700

Definición en la línea **37** del archivo **sketch_may7a.ino**.

#define SOIL_THRESHOLD_HIGH 80.0

Definición en la línea **36** del archivo **sketch_may7a.ino**.

#define SOIL_THRESHOLD_LOW 30.0

Definición en la línea **35** del archivo **sketch_may7a.ino**.

#define TEMP_THRESHOLD 27.0

Umbrales predefinidos para generar alertas.

Definición en la línea **33** del archivo **sketch_may7a.ino**.

Documentación de funciones

UniversalTelegramBot bot (BOTtoken , client)

void DHTTask (void * parameter)

Tarea que procesa los datos del sensor DHT.

Revisa si la temperatura o la humedad están por encima de los umbrales y en ese caso genera un mensaje para enviar.

Definición en la línea **85** del archivo **sketch_may7a.ino**.

void LDRTask (void * parameter)

Tarea que procesa los datos del sensor LDR (luz).

Mensajes de alerta para los umbrales de luz.

Definición en la línea **128** del archivo **sketch_may7a.ino**.

void loop ()

Bucle principal del programa (no hace nada porque todo se gestiona con tareas).

Definición en la línea **228** del archivo **sketch_may7a.ino**.

void OnDataRecv (const esp_now_recv_info_t * info, const uint8_t * incoming, int len)

Callback llamado automáticamente por ESP-NOW al recibir datos.

Clasifica los datos entrantes según la MAC de origen y los coloca en la cola correspondiente.

Parámetros

info	Información sobre la recepción, incluyendo la MAC de origen
incoming	Puntero a los datos recibidos
len	Longitud de los datos recibidos

Definición en la línea **64** del archivo **sketch_may7a.ino**.

void setup ()

Función principal de inicialización.

Configura la conexión WiFi, sincroniza hora, inicializa ESP-NOW, registra los peers emisores, crea las colas y tareas.

Instancia de colas para paquetes de Datos provenientes de los nodos emisores.
Creación de Tareas para la lectura de datos de los sensores.
Definición en la línea **170** del archivo **sketch_may7a.ino**.

void SoilTask (void * parameter)

Tarea que procesa los datos del sensor de humedad de suelo.

Mensajes de alerta para los umbrales de Humedad del suelo.
Definición en la línea **105** del archivo **sketch_may7a.ino**.

void TelegramTask (void * parameter)

Tarea dedicada al envío de mensajes por Telegram.
Espera mensajes de las otras tareas y los envía usando el bot configurado.
Mensajes de alerta cuando los mensajes se envían o no a telegram.
Definición en la línea **151** del archivo **sketch_may7a.ino**.

Documentación de variables

WiFiClientSecure client

Definición en la línea **29** del archivo **sketch_may7a.ino**.

QueueHandle_t dhtQueue

Colas para manejar los datos recibidos por cada tipo de sensor.
Definición en la línea **50** del archivo **sketch_may7a.ino**.

uint8_t dhtSenderMAC[] = {0x08, 0xD1, 0xF9, 0xEE, 0x0A, 0x04}

MAC addresses de los nodos emisores.
Definición en la línea **40** del archivo **sketch_may7a.ino**.

QueueHandle_t ldrQueue

Definición en la línea **52** del archivo **sketch_may7a.ino**.

uint8_t ldrSenderMAC[] = {0xa0, 0xa3, 0xb3, 0x2a, 0xE0, 0x20}

Definición en la línea **42** del archivo **sketch_may7a.ino**.

QueueHandle_t msgQueue

Definición en la línea **53** del archivo **sketch_may7a.ino**.

const char* password = "25310tati"

Definición en la línea **23** del archivo **sketch_may7a.ino**.

QueueHandle_t soilQueue

Definición en la línea **51** del archivo **sketch_may7a.ino**.

uint8_t soilSenderMAC[] = {0xa0, 0xb7, 0x65, 0x1b, 0x18, 0x50}

Definición en la línea **41** del archivo **sketch_may7a.ino**.

const char* ssid = "iPhone"

Libresias a utilizar para cada uno de los componentes.

Credenciales WiFi para conectar el receptor a internet

Definición en la línea **22** del archivo **sketch_may7a.ino**.

sketch_may7a.ino

Ir a la documentación de este archivo.

```
00001
00009 #ifdef ESP32
00010   #include <WiFi.h>
00011 #else
00012   #include <ESP8266WiFi.h>
00013 #endif
00014
00015 #include <esp_now.h>
00016 #include <WiFiClientSecure.h>
00017 #include <UniversalTelegramBot.h>
00018 #include <time.h>
00019 #include "esp_wifi.h"
00020
00022 const char* ssid     = "iPhone";
00023 const char* password = "25310tati";
00024
00026 #define BOTtoken  "7672713502:AAH55Vyi1rU-EF1eij3dnpqYvRMLxzsE58g"
00027 #define CHAT_ID   "5365006160"
00028
00029 WiFiClientSecure client;
00030 UniversalTelegramBot bot(BOTtoken, client);
00031
00033 #define TEMP_THRESHOLD      27.0
00034 #define HUM_THRESHOLD       80.0
00035 #define SOIL_THRESHOLD_LOW  30.0
00036 #define SOIL_THRESHOLD_HIGH 80.0
00037 #define LDR_THRESHOLD       700
00038
00040 uint8_t dhtSenderMAC[]  = {0x08, 0xD1, 0xF9, 0xEE, 0x0A, 0x04};
00041 uint8_t soilSenderMAC[] = {0xa0, 0xb7, 0x65, 0x1b, 0x18, 0x50};
00042 uint8_t ldrSenderMAC[]  = {0xa0, 0xa3, 0xb3, 0x2a, 0xE0, 0x20};
00043
00045 typedef struct { float temp, hum; } dht_message_t;
00046 typedef struct { float soil; }      soil_message_t;
00047 typedef struct { int ldrValue; }    light_message_t;
00048
00050 QueueHandle_t dhtQueue;
00051 QueueHandle_t soilQueue;
00052 QueueHandle_t ldrQueue;
00053 QueueHandle_t msgQueue;
00054
00064 void OnDataRecv(const esp_now_recv_info_t *info, const uint8_t *incoming, int len)
{
00065   if (memcmp(info->src_addr, dhtSenderMAC, 6) == 0 && len == sizeof(dht_message_t))
{
00066     dht_message_t d;
00067     memcpy(&d, incoming, len);
00068     xQueueSend(dhtQueue, &d, portMAX_DELAY);
00069   } else if (memcmp(info->src_addr, soilSenderMAC, 6) == 0 && len ==
sizeof(soil_message_t)) {
00070     soil_message_t s;
00071     memcpy(&s, incoming, len);
00072     xQueueSend(soilQueue, &s, portMAX_DELAY);
00073   } else if (memcmp(info->src_addr, ldrSenderMAC, 6) == 0 && len ==
sizeof(light_message_t)) {
00074     light_message_t l;
00075     memcpy(&l, incoming, len);
00076     xQueueSend(ldrQueue, &l, portMAX_DELAY);
00077   }
00078 }
00079
00085 void DHTTask(void *parameter) {
00086   dht_message_t d;
00087   for (;;) {
00088     if (xQueueReceive(dhtQueue, &d, portMAX_DELAY)) {
00089       String msg = "";
00090       Serial.printf("DHT: T=%.2f°C, H=%.2f%%\n", d.temp, d.hum);
00091       if (d.temp > TEMP_THRESHOLD) msg += "âš  Temp alta: " + String(d.temp, 2) +
"Â°C\n";
```

```
00092       if (d.hum  > HUM_THRESHOLD)  msg += "ðŸ’§ Hum alta:  " + String(d.hum, 2) +
"%\n";
00093       if (msg.length()) {
00094         char* buf = (char*)malloc(msg.length() + 1);
00095         msg.toCharArray(buf, msg.length() + 1);
00096         xQueueSend(msgQueue, &buf, portMAX_DELAY);
00097       }
00098     }
00099   }
00100 }
00101
00105 void SoilTask(void *parameter) {
00106   soil_message_t s;
00107   for (;;) {
00108     if (xQueueReceive(soilQueue, &s, portMAX_DELAY)) {
00109       String msg = "";
00110       Serial.printf("Suelo: %.1f%%\n", s.soil);
00114       if      (s.soil < SOIL_THRESHOLD_LOW)  msg += "ðŸŒ± Suelo muy seco: " +
String(s.soil, 1) + "%\n";
00115       else if (s.soil > SOIL_THRESHOLD_HIGH) msg += "ðŸ’§ Suelo muy hÃºmedo: " +
String(s.soil, 1) + "%\n";
00116       if (msg.length()) {
00117         char* buf = (char*)malloc(msg.length() + 1);
00118         msg.toCharArray(buf, msg.length() + 1);
00119         xQueueSend(msgQueue, &buf, portMAX_DELAY);
00120       }
00121     }
00122   }
00123 }
00124
00128 void LDRTask(void *parameter) {
00129   light_message_t l;
00130   for (;;) {
00131     if (xQueueReceive(ldrQueue, &l, portMAX_DELAY)) {
00132       String msg = "";
00133       Serial.printf("LDR: %d\n", l.ldrValue);
00137       if (l.ldrValue > LDR_THRESHOLD) msg += "ðŸŒ‘ Luz baja: " + String(l.ldrValue)
+ "\n";
00138       else                            msg += "ðŸ’¡ Luz normal: " +
String(l.ldrValue) + "\n";
00139       char* buf = (char*)malloc(msg.length() + 1);
00140       msg.toCharArray(buf, msg.length() + 1);
00141       xQueueSend(msgQueue, &buf, portMAX_DELAY);
00142     }
00143   }
00144 }
00145
00151 void TelegramTask(void *parameter) {
00152   char* recvBuffer;
00153   for (;;) {
00154     if (xQueueReceive(msgQueue, &recvBuffer, portMAX_DELAY)) {
00155       bool ok = bot.sendMessage(CHAT_ID, String(recvBuffer), "");
00159       Serial.println(ok ? "â… Telegram enviado" : "â Error en Telegram");
00160       free(recvBuffer);
00161     }
00162   }
00163 }
00164
00170 void setup() {
00171   Serial.begin(115200);
00172   WiFi.mode(WIFI_STA);
00173   WiFi.begin(ssid, password);
00174   Serial.print("Conectando WiFi");
00175   while (WiFi.status() != WL_CONNECTED) {
00176     delay(500); Serial.print(".");
00177   }
00178   Serial.println("\nConectado IP: " + WiFi.localIP().toString());
00179
00180   int canal = WiFi.channel();
00181   Serial.printf("Canal: %d\n", canal);
00182   configTime(-5 * 3600, 0, "pool.ntp.org", "time.nist.gov");
00183   struct tm ti;
00184   while (!getLocalTime(&ti)) delay(200);
00185   Serial.printf("Hora: %02d:%02d:%02d\n", ti.tm_hour, ti.tm_min, ti.tm_sec);
00186
00187   client.setInsecure();
00188   bot.sendMessage(CHAT_ID, "ðŸ¤– Bot receptor iniciado", "");
```

```
00192   dhtQueue  = xQueueCreate(5, sizeof(dht_message_t));
00193   soilQueue = xQueueCreate(5, sizeof(soil_message_t));
00194   ldrQueue  = xQueueCreate(5, sizeof(light_message_t));
00195   msgQueue  = xQueueCreate(5, sizeof(char*));
00199   xTaskCreate(DHTTask,  "DHTTask",  4096, NULL, 1, NULL);
00200   xTaskCreate(SoilTask, "SoilTask", 4096, NULL, 1, NULL);
00201   xTaskCreate(LDRTask,  "LDRTask",  4096, NULL, 1, NULL);
00202   xTaskCreate(TelegramTask, "TelegramTask", 8192, NULL, 1, NULL);
00203
00204   esp_wifi_set_channel(canal, WIFI_SECOND_CHAN_NONE);
00205   if (esp_now_init() != ESP_OK) {
00206     Serial.println("â□Œ Error al iniciar ESP-NOW");
00207     return;
00208   }
00209   esp_now_register_recv_cb(OnDataRecv);
00210
00211   esp_now_peer_info_t peer = {};
00212   peer.channel = canal;
00213   peer.encrypt = false;
00214
00215   memcpy(peer.peer_addr, dhtSenderMAC, 6);
00216   esp_now_add_peer(&peer);
00217
00218   memcpy(peer.peer_addr, soilSenderMAC, 6);
00219   esp_now_add_peer(&peer);
00220
00221   memcpy(peer.peer_addr, ldrSenderMAC, 6);
00222   esp_now_add_peer(&peer);
00223 }
00224
00228 void loop() {
00229   vTaskDelay(portMAX_DELAY);
00230 }
```