MCGA

NODO SOIL_PUMP

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 para empaquetar y enviar valores de sensores.

Atributos públicos

- float **soil**

Descripción detallada

Estructura para empaquetar y enviar valores de sensores.

Por simplicidad sólo incluye humedad de suelo en este nodo.

Definición en la línea **65** del archivo **NodoSoil.ino**.

Documentación de datos miembro

float struct_message_send::soil

Porcentaje de humedad de suelo calculado.

Definición en la línea **66** del archivo **NodoSoil.ino**.

La documentación de esta estructura está generada del siguiente archivo:

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

Documentación de archivos

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

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

Clases

struct **struct_message_send***Estructura para empaquetar y enviar valores de sensores.*

defines

- #define **SOIL_PIN** 34
  Pin analógico conectado al sensor de humedad de suelo.

- #define **RELAY_PIN** 23
  Pin digital conectado al relé que controla la bomba.

- #define **UMBRAL_HUMEDAD** 30
  Porcentaje de humedad mínimo antes de activar la bomba.

- #define **VALOR_SECO** 2590
  Lectura ADC correspondiente a suelo completamente seco.

- #define **VALOR_MOJADO** 1200
  Lectura ADC correspondiente a suelo completamente mojado.

- #define **RELAY_ACTIVO_LOW** false
  Indicador de lógica activa del relé.

Funciones

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

- void **tareaHumedad** (void *parameter)
  Tarea FreeRTOS que lee el sensor de humedad, envía vía ESP-NOW, activa la bomba si es necesario y entra en deep sleep.

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

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

Variables

- uint8_t **receiverMAC** [] = {0x5C, 0x01, 0x3B, 0x72, 0xF2, 0xCC}
  Dirección MAC del receptor ESP32 (Nodo principal) para ESP-NOW.

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

- volatile bool **envioCompleto** = false
  Variable volatile para indicar cuando el envío ESP-NOW ha terminado.

Documentación de «define»

#define RELAY_ACTIVO_LOW false

Indicador de lógica activa del relé.

false = relé se activa escribiendo HIGH en el pin; true = se activa con LOW.

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

#define RELAY_PIN 23

Pin digital conectado al relé que controla la bomba.

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

#define SOIL_PIN 34

Pin analógico conectado al sensor de humedad de suelo.

Se utiliza ADC1_CH6, mapeado al GPIO34 en ESP32.

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

#define UMBRAL_HUMEDAD 30

Porcentaje de humedad mínimo antes de activar la bomba.

Si la humedad medida (%), calculada entre seco y mojado, es menor que este umbral, la bomba se enciende durante 2 segundos.

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

#define VALOR_MOJADO 1200

Lectura ADC correspondiente a suelo completamente mojado.

Valor aproximado obtenido en calibración: 1200 (100% humedad).

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

#define VALOR_SECO 2590

Lectura ADC correspondiente a suelo completamente seco.

Valor aproximado obtenido en calibración: 2590 (0% humedad).

Definición en la línea **40** del archivo **NodoSoil.ino**.

Documentación de funciones

void loop ()

Función loop de Arduino.

Nota

Vacía: toda la lógica se ejecuta en "tareaHumedad" y deep sleep.

Definición en la línea **218** del archivo **NodoSoil.ino**.

void OnDataSent (const uint8_t * mac_addr, esp_now_send_status_t status)

Callback invocado tras intentar enviar un paquete ESP-NOW.

Parámetros

mac_addr	Dirección MAC destino.
status	Resultado del envío (éxito o fallo).

Muestra en el monitor serial si el paquete se envió correctamente y actualiza la bandera envioCompleto.

Definición en la línea **90** del archivo **NodoSoil.ino**.

void setup ()

Función setup principal de Arduino.

Inicializa Serial, configura pines, modo WiFi y ESP-NOW, registra callback de envío y crea la tarea de humedad.

Definición en la línea **164** del archivo **NodoSoil.ino**.

void tareaHumedad (void * parameter)

Tarea FreeRTOS que lee el sensor de humedad, envía vía ESP-NOW, activa la bomba si es necesario y entra en deep sleep.

Parámetros

parameter	Parámetros pasados a la tarea (no utilizado, NULL).

1) Lee valor raw del ADC y lo mapea a porcentaje 0–100%. 2) Empaqueta y envía el porcentaje al nodo receptor. 3) Espera confirmación de envío (timeout = 500 ms). 4) Si humedad < UMBRAL_HUMEDAD, enciende la bomba 2 segundos. 5) Apaga bomba y activa deep sleep por 30 s.

Definición en la línea **107** del archivo **NodoSoil.ino**.

Documentación de variables

volatile bool envioCompleto = false

Variable volatile para indicar cuando el envío ESP-NOW ha terminado.
Se marca a true en la función callback **OnDataSent()**.
Definición en la línea **79** del archivo **NodoSoil.ino**.

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

Dirección MAC del receptor ESP32 (Nodo principal) para ESP-NOW.
Definición en la línea **57** del archivo **NodoSoil.ino**.

struct_message_send sensorSend

Variable global que almacena la estructura a enviar.
Definición en la línea **72** del archivo **NodoSoil.ino**.

NodoSoil.ino

Ir a la documentación de este archivo.

```
00001
00011
00012 #include <WiFi.h>
00013 #include <esp_now.h>
00014 #include "esp_wifi.h"
00015 #include "esp_sleep.h"
00016
00017 // ── Configuración de pines y constantes ──
00022 #define SOIL_PIN        34
00023
00027 #define RELAY_PIN       23
00028
00034 #define UMBRAL_HUMEDAD  30
00035
00040 #define VALOR_SECO      2590
00041
00046 #define VALOR_MOJADO    1200
00047
00052 #define RELAY_ACTIVO_LOW false
00053
00057 uint8_t receiverMAC[] = {0x5C, 0x01, 0x3B, 0x72, 0xF2, 0xCC};
00058
00059 // ── Estructura de datos a enviar ──
00065 typedef struct {
00066   float soil;
00067 } struct_message_send;
00068
00072 struct_message_send sensorSend;
00073
00074 // ── Flag de envío completado ──
00079 volatile bool envioCompleto = false;
00080
00081 // ── Callback de envío ──
00090 void OnDataSent(const uint8_t *mac_addr, esp_now_send_status_t status) {
00091   Serial.printf("ðŸ"¡ EnvÃ-o ESP-NOW: %s\n", status == ESP_NOW_SEND_SUCCESS ? "OK"
: "FALLÃ"");
00092   envioCompleto = true;
00093 }
00094
00095 // ── Tarea de lectura de humedad ──
00107 void tareaHumedad(void *parameter) {
00108   // Lectura cruda del sensor
00109   int raw = analogRead(SOIL_PIN);
00110   // Mapeo lineal de valor ADC a porcentaje
00111   int pct = map(raw, VALOR_SECO, VALOR_MOJADO, 0, 100);
00112   pct = constrain(pct, 0, 100);
00113   Serial.printf("ðŸŒ± Humedad: RAW = %d â†' %d%%\n", raw, pct);
00114
00115   // Preparar datos y enviar
00116   sensorSend.soil = pct;
00117   envioCompleto = false;
00118   esp_err_t res = esp_now_send(receiverMAC, (uint8_t *)&sensorSend,
sizeof(sensorSend));
00119   if (res != ESP_OK) {
00120     Serial.printf("â□Œ Error al enviar ESP-NOW: %d\n", res);
00121   }
00122
00123   // Espera activa con timeout de 500 ms
00124   unsigned long t0 = millis();
00125   while (!envioCompleto && millis() - t0 < 500) {
00126     vTaskDelay(pdMS_TO_TICKS(10));
00127   }
00128
00129   // Control de bomba según humedad
00130   if (pct < UMBRAL_HUMEDAD) {
00131     Serial.println("ðŸ'§ Suelo seco: Activando bomba...");
00132     // Nivel lógico según configuración de relé
00133     if (RELAY_ACTIVO_LOW)
00134       digitalWrite(RELAY_PIN, LOW);
00135     else
```

```
00136       digitalWrite(RELAY_PIN, HIGH);
00137
00138     // Mantener bomba encendida 2 segundos
00139     vTaskDelay(pdMS_TO_TICKS(2000));
00140
00141     // Apagar bomba
00142     if (RELAY_ACTIVO_LOW)
00143       digitalWrite(RELAY_PIN, HIGH);
00144     else
00145       digitalWrite(RELAY_PIN, LOW);
00146
00147     Serial.println("ðŸš¿ Bomba apagada.");
00148   } else {
00149     Serial.println("â… Suelo hÃ°medo, no se activa bomba.");
00150   }
00151
00152   // Prepararse para deep sleep
00153   Serial.println("ðŸ˜´ Entrando en deep sleep...");
00154   vTaskDelay(pdMS_TO_TICKS(100));
00155   esp_deep_sleep_start();
00156 }
00157
00164 void setup() {
00165   Serial.begin(115200);
00166   delay(100);
00167
00168   // Configuración de GPIO para relé
00169   pinMode(RELAY_PIN, OUTPUT);
00170   // Asegurar estado de apagado lógico
00171   if (RELAY_ACTIVO_LOW)
00172     digitalWrite(RELAY_PIN, HIGH);
00173   else
00174     digitalWrite(RELAY_PIN, LOW);
00175
00176   // Programar timer para deep sleep de 30 segundos
00177   esp_sleep_enable_timer_wakeup(30 * 1000000ULL);
00178
00179   // Modo WiFi estación y fijar canal para ESP-NOW
00180   WiFi.mode(WIFI_STA);
00181   esp_wifi_set_channel(6, WIFI_SECOND_CHAN_NONE);
00182
00183   // Inicializar ESP-NOW
00184   if (esp_now_init() != ESP_OK) {
00185     Serial.println("â□Œ Error al iniciar ESP-NOW");
00186     return;
00187   }
00188
00189   // Registrar callback de envío
00190   esp_now_register_send_cb(OnDataSent);
00191
00192   // Añadir peer (nodo receptor)
00193   esp_now_peer_info_t peerInfo = {};
00194   memcpy(peerInfo.peer_addr, receiverMAC, 6);
00195   peerInfo.channel = 6;
00196   peerInfo.encrypt = false;
00197   if (esp_now_add_peer(&peerInfo) != ESP_OK) {
00198     Serial.println("â□Œ Error al agregar receptor");
00199     return;
00200   }
00201
00202   // Crear tarea libreRTOS para gestión de humedad
00203   xTaskCreatePinnedToCore(
00204     tareaHumedad,
00205     "TareaHumedad",
00206     4096,
00207     NULL,
00208     1,
00209     NULL,
00210     1
00211   );
00212 }
00213
00218 void loop() {
00219   // No se usa, queda en standby
00220 }
```