#include <stdio.h>

#include <string.h>

#include "freertos/FreeRTOS.h"

#include "freertos/task.h"

#include "driver/uart.h"

#include "esp\_log.h"

#include "esp\_wifi.h"

#include "esp\_http\_client.h"

#include "driver/gpio.h"

#include "lora.h"

// Wi-Fi Credentials

#define WIFI\_SSID "IoT"

#define WIFI\_PASSWORD "IOT@123456789"

// ThingSpeak API Key

#define THINGSPEAK\_WRITE\_API\_KEY "UKQI7N3GYO53BSQG"

// UART Configuration

#define UART\_NUM UART\_NUM\_1

#define BUF\_SIZE (1024)

#define TXD\_PIN (GPIO\_NUM\_17)

#define RXD\_PIN (GPIO\_NUM\_16)

// LoRa Configuration

#define LORA\_SCK 5

#define LORA\_MISO 19

#define LORA\_MOSI 27

#define LORA\_CS 18

#define LORA\_RST 14

#define LORA\_IRQ 26

#define LORA\_FREQUENCY 868E6

static const char \*TAG = "ESP32\_Main";

// Wi-Fi Event Handler

void wifi\_event\_handler(void\* arg, esp\_event\_base\_t event\_base,

int32\_t event\_id, void\* event\_data) {

if (event\_base == WIFI\_EVENT && event\_id == WIFI\_EVENT\_STA\_START) {

esp\_wifi\_connect();

} else if (event\_base == WIFI\_EVENT && event\_id == WIFI\_EVENT\_STA\_DISCONNECTED) {

ESP\_LOGI(TAG, "Reconnecting to Wi-Fi...");

esp\_wifi\_connect();

} else if (event\_base == IP\_EVENT && event\_id == IP\_EVENT\_STA\_GOT\_IP) {

ESP\_LOGI(TAG, "Connected to Wi-Fi.");

}

}

// Initialize Wi-Fi

void init\_wifi(void) {

esp\_netif\_init();

esp\_event\_loop\_create\_default();

esp\_netif\_create\_default\_wifi\_sta();

wifi\_init\_config\_t cfg = WIFI\_INIT\_CONFIG\_DEFAULT();

esp\_wifi\_init(&cfg);

esp\_event\_handler\_register(WIFI\_EVENT, ESP\_EVENT\_ANY\_ID, &wifi\_event\_handler, NULL);

esp\_event\_handler\_register(IP\_EVENT, IP\_EVENT\_STA\_GOT\_IP, &wifi\_event\_handler, NULL);

wifi\_config\_t wifi\_config = {

.sta = {

.ssid = WIFI\_SSID,

.password = WIFI\_PASSWORD

},

};

esp\_wifi\_set\_mode(WIFI\_MODE\_STA);

esp\_wifi\_set\_config(WIFI\_IF\_STA, &wifi\_config);

esp\_wifi\_start();

}

// HTTP POST to ThingSpeak

void send\_to\_thingspeak(const char\* data) {

char post\_data[256];

sprintf(post\_data, "api\_key=%s&field1=%s", THINGSPEAK\_WRITE\_API\_KEY, data);

esp\_http\_client\_config\_t config = {

.url = "http://api.thingspeak.com/update",

};

esp\_http\_client\_handle\_t client = esp\_http\_client\_init(&config);

esp\_http\_client\_set\_method(client, HTTP\_METHOD\_POST);

esp\_http\_client\_set\_header(client, "Content-Type", "application/x-www-form-urlencoded");

esp\_http\_client\_set\_post\_field(client, post\_data, strlen(post\_data));

esp\_err\_t err = esp\_http\_client\_perform(client);

if (err == ESP\_OK) {

ESP\_LOGI(TAG, "ThingSpeak Response: %d", esp\_http\_client\_get\_status\_code(client));

} else {

ESP\_LOGE(TAG, "HTTP POST failed: %s", esp\_err\_to\_name(err));

}

esp\_http\_client\_cleanup(client);

}

// UART Task

void uart\_task(void \*arg) {

uint8\_t data[BUF\_SIZE];

while (1) {

int len = uart\_read\_bytes(UART\_NUM, data, BUF\_SIZE - 1, 20 / portTICK\_RATE\_MS);

if (len > 0) {

data[len] = '\0'; // Null-terminate the received data

ESP\_LOGI(TAG, "Received Data: %s", data);

// Send to ThingSpeak

send\_to\_thingspeak((const char \*)data);

// Send via LoRa

lora\_send\_packet(data, len);

}

}

}

// Main Function

void app\_main(void) {

// Initialize peripherals

init\_wifi();

// Initialize UART

uart\_config\_t uart\_config = {

.baud\_rate = 9600,

.data\_bits = UART\_DATA\_8\_BITS,

.parity = UART\_PARITY\_DISABLE,

.stop\_bits = UART\_STOP\_BITS\_1,

.flow\_ctrl = UART\_HW\_FLOWCTRL\_DISABLE,

.source\_clk = UART\_SCLK\_APB,

};

uart\_param\_config(UART\_NUM, &uart\_config);

uart\_set\_pin(UART\_NUM, TXD\_PIN, RXD\_PIN, UART\_PIN\_NO\_CHANGE, UART\_PIN\_NO\_CHANGE);

uart\_driver\_install(UART\_NUM, BUF\_SIZE, 0, 0, NULL, 0);

// Initialize LoRa

lora\_init();

lora\_set\_frequency(LORA\_FREQUENCY);

lora\_enable\_crc();

// Start UART task

xTaskCreate(uart\_task, "uart\_task", 2048, NULL, 10, NULL);

}