Anexo 2

Código del dispositivo para la medición de la calidad de la señal LoRa

Daria Bakteeva Díaz-Piñeiro– Febrero 2024

```
#include <Arduino.h>
#include "lmic_project_config.h"
#include <lmic.h>
#include <hal/hal.h>
#include <U8g2lib.h>
#include "images.h"
#include <Thread.h>
#include <ThreadController.h>

// OLED Display Object
U8G2_SSD1306_128X64_NONAME_F_HW_I2C u8g2(U8G2_R0, /* reset=*/
U8X8_PIN_NONE);


// Global Variables
const int botonPin = 25; // Button pin
int ScreenState = 1; // State of the screen (on/off)
int Button = 1; // Button state
int Radio = 0; // Radio state for LoRa operations
osjob_t sendjob; // Job for sending data


// Animation Variables
int ballX = 64; // Horizontal position of the ball
int ballY = 32; // Vertical position of the ball
int ballRadius = 5; // Radius of the ball
int ballSpeedX = 4; // Horizontal speed
int ballSpeedY = 4; // Vertical speed
unsigned long lastAnimationUpdate = 0; // Time of last animation update
const int animationInterval = 5; // Interval between animation updates in
ms


// LoRaWAN Configuration for TTGO LoRa32 OLED V1.3
const lmic_pinmap lmic_pins = {
    .nss = 18,
    .rxtx = LMIC_UNUSED_PIN,
    .rst = 14,
    .dio = {26, 33, 32},  // DIO0, DIO1, DIO2
};


// OTAA Keys
static const PROGMEM u1_t APPEUI[8] = {0x77, 0x77, 0x77, 0x77, 0x77,
0x77, 0x77, 0x77};
static const PROGMEM u1_t DEVEUI[8] = {0xAE, 0x2A, 0x06, 0xD0, 0x7E,
0xD5, 0xB3, 0x70};
static const PROGMEM u1_t APPKEY[16] = {0x8B, 0x5F, 0xA9, 0xA9, 0x91,
0x46, 0x92, 0x2F, 0x66, 0xF1, 0x50, 0x98, 0xC0, 0x61, 0x0F, 0x09};
```

```
// LMIC Event Callbacks
void os_getArtEui (u1_t* buf) { memcpy(buf, APPEUI, 8); }

void os_getDevEui (u1_t* buf) { memcpy(buf, DEVEUI, 8); }

void os_getDevKey (u1_t* buf) { memcpy(buf, APPKEY, 16); }


// Event Handling for LoRaWAN
void onEvent(ev_t ev) {
    Serial.print(os_getTime());
    Serial.print(": ");
    switch(ev) {
        case EV_JOINED: // Once joined the network
            Serial.println(F("EV_JOINED"));
            LMIC_setLinkCheckMode(0); // Disable periodic joins
            break;

        case EV_TXCOMPLETE: // After transmission is complete
            Serial.println(F("EV_TXCOMPLETE"));
            if (LMIC.txrxFlags & TXRX_ACK) {
                Serial.println(F("ACK Received"));
                int rssi = LMIC.rssi-72;
                float snr = (LMIC.snr)/4.0;

                u8g2.clearBuffer();
                u8g2.setFont(u8g2_font_ncenB10_tr);
                u8g2.drawStr(0, 24, "RSSI:");
                u8g2.drawStr(64, 24, String(rssi).c_str());
                u8g2.drawStr(0, 48, "SNR:");
                u8g2.drawStr(64, 48, String(snr).c_str());
                u8g2.sendBuffer();
            }
            Radio = 0;
            break;

    }
}

// Function to Send Data
void do_send() {
    if (!(LMIC.opmode & OP_TXRXPEND)) {
        uint8_t data[] = {0x01};
        LMIC_setTxData2(1, data, sizeof(data), 1);
        Serial.println(F("Sending..."));
    } else {
        Serial.println(F("Operation pending"));
    }
```

```
}

// Function to Display Loading Animation
void displayLoadingAnimation() {
    if (millis() - lastAnimationUpdate > animationInterval) {
        lastAnimationUpdate = millis();

        // Update ball position
        ballX += ballSpeedX;
        ballY += ballSpeedY;

        // Boundary collision check
        if (ballX <= ballRadius || ballX >= 128 - ballRadius) {
            ballSpeedX = -ballSpeedX; // Reverse horizontal direction
        }
        if (ballY <= ballRadius || ballY >= 64 - ballRadius) {
            ballSpeedY = -ballSpeedY; // Reverse vertical direction
        }

        // Draw the ball and loading text
        u8g2.clearBuffer();
        u8g2.drawDisc(ballX, ballY, ballRadius); // Filled circle
        u8g2.setFont(u8g2_font_ncenB08_tr);
        u8g2.drawStr(40, 40, "Loading...");
        u8g2.sendBuffer();
    }
}

// Thread Control
ThreadController controll = ThreadController();
Thread* ScreenThread = new Thread();
Thread* RadioThread = new Thread();
Thread* ButtonThread = new Thread();

// Functions for Threads
void ScreenFunc() {
    if (ScreenState == 1 && Button == 1) {
        displayLoadingAnimation();
    }
}


void ButtonFunc() {
    const unsigned long debounceDelay = 50; // Debounce delay in
milliseconds
    static unsigned long lastDebounceTime = 0;
    static int lastButtonState = HIGH; // Last state of the button

    int reading = digitalRead(botonPin);
```

```
    if ((millis() - lastDebounceTime) > debounceDelay) {
        if (reading != lastButtonState) {
            Serial.println("Reading button.");
            lastDebounceTime = millis();

            if (reading == LOW) {
                Serial.println("Button pressed.");
                Button = 0; // Set Button as pressed
            }
        }
    }
    lastButtonState = reading;
}

void RadioFunc() {
    if (Button == 0 && Radio == 0) {
        if (!(LMIC.opmode & OP_JOINING) && LMIC.devaddr == 0) {
            Serial.println("Starting join process...");
            LMIC_startJoining();
        } else if (LMIC.devaddr != 0) {
            Serial.println("Button pressed, sending data...");
            do_send();
        }
    }
    Radio = 1; // Reset Radio state
    Button = 1; // Reset Button state
}

// Setup Function
void setup() {
    Serial.begin(115200);
    u8g2.begin();
    u8g2.clearBuffer();
    u8g2.drawXBM(1, 1, logo_medialab_width, logo_medialab_height,
medialab_bits);
    u8g2.sendBuffer();
    delay(3000);

    pinMode(botonPin, INPUT_PULLDOWN);

    os_init();
    LMIC_reset();

    // Frequency band configuration
    #ifdef CFG_eu868
    LMIC_setupChannel(0, 868100000, DR_RANGE_MAP(DR_SF12,
DR_SF7),  BAND_CENTI);
```

```
    LMIC_setupChannel(1, 868300000, DR_RANGE_MAP(DR_SF12, DR_SF7B),
BAND_CENTI);
    LMIC_setupChannel(2, 868500000, DR_RANGE_MAP(DR_SF12,
DR_SF7),  BAND_CENTI);
    LMIC_setupChannel(3, 867100000, DR_RANGE_MAP(DR_SF12,
DR_SF7),  BAND_CENTI);
    LMIC_setupChannel(4, 867300000, DR_RANGE_MAP(DR_SF12,
DR_SF7),  BAND_CENTI);
    LMIC_setupChannel(5, 867500000, DR_RANGE_MAP(DR_SF12,
DR_SF7),  BAND_CENTI);
    LMIC_setupChannel(6, 867700000, DR_RANGE_MAP(DR_SF12,
DR_SF7),  BAND_CENTI);
    LMIC_setupChannel(7, 867900000, DR_RANGE_MAP(DR_SF12,
DR_SF7),  BAND_CENTI);
    LMIC_setupChannel(8, 868800000,
DR_RANGE_MAP(DR_FSK,  DR_FSK),  BAND_MILLI);

    #endif

    LMIC_setDrTxpow(DR_SF7,14); // Set SF7 as spreading factor and 14 dBm
as transmission power

    Serial.println("Setup complete.");

    // Setting up threads
    ScreenThread->onRun(ScreenFunc);
    ScreenThread->setInterval(animationInterval);
    ButtonThread->onRun(ButtonFunc);
    ButtonThread->setInterval(50);
    RadioThread->onRun(RadioFunc);
    RadioThread->setInterval(100);

    // Adding threads to controller
    controll.add(ScreenThread);
    controll.add(ButtonThread);
    controll.add(RadioThread);
}

// Main Loop
void loop() {
    controll.run(); // Run thread controller
}
```