/\*

#include <Arduino.h>

#include <WiFi.h>

#include <WebServer.h> // HTTP/HTTPS-сервер для ESP32

#include <BearSSLHelpers.h> // Додаємо BearSSL для HTTPS

#include "arduinoFFT.h"

#include <FastLED.h>

#define SAMPLES 128 // до блоку обробки FFT - кількість зразків звуку (кадрів, семплів), взятих протягом 1 секунди

#define SAMPLING\_FREQ 10000 // частота дискретизації - довжина 1 зразка (семпла, кадра) у мікросекундах

#define LED\_PIN 2 // пін, на якому led-кружальце

#define NUM\_LEDS 16 // кількість led-ів у кружальці

const char\* ssid = "Vidrodgennya\_2G";

const char\* password = "960010Kalash";

// Самопідписаний сертифікат і ключ - згенеровані OpenSSL локально

const char\* cert = \

"-----BEGIN CERTIFICATE-----\n" \

"MIIDmzCCAoOgAwIBAgIUbZoRcZUSBigVfkSaE1Yk1wMBJ/cwDQYJKoZIhvcNAQEL\n" \

"BQAwXTELMAkGA1UEBhMCVUExEzARBgNVBAgMClNvbWUtU3RhdGUxITAfBgNVBAoM\n" \

"GEludGVybmV0IFdpZGdpdHMgUHR5IEx0ZDEWMBQGA1UEAwwNMTkyLjE2OC4wLjEw\n" \

"MTAeFw0yNTAzMDQyMTQxMDhaFw0yNjAzMDQyMTQxMDhaMF0xCzAJBgNVBAYTAlVB\n" \

"MRMwEQYDVQQIDApTb21lLVN0YXRlMSEwHwYDVQQKDBhJbnRlcm5ldCBXaWRnaXRz\n" \

"IFB0eSBMdGQxFjAUBgNVBAMMDTE5Mi4xNjguMC4xMDEwggEiMA0GCSqGSIb3DQEB\n" \

"AQUAA4IBDwAwggEKAoIBAQDiOMss0W6omn0WJEur6/+EgnrwmaqMCpDHgLj7xlzF\n" \

"Hhw3PyyGPUt+UTB95w2dmQtzM+36RDuSTYa9w/4p4eZ8cqzyDXK3fD9tz7Db011L\n" \

"n9yZYujGqcsKi6PPYlJWYk/UuV6rggyfTx4b1ioF/xkL+cQ2OvqinBZLbTBcojkS\n" \

"WTNRvnveQ4AMLghszdk62pmYTzeZeN2Cx06+0Kxit8PC0y/SG0rF+P0tYZiZvtMK\n" \

"2Cj5GCiSUBX3aRJNrpbga+9Vir2OZmxWzMeZrgtQ50/Ji3YAbDZXdBo/t6D6ECsr\n" \

"ZLbtoaTS6nz9UQRooPjUzNl1tdONVuiLkRoo4vsr+vu3AgMBAAGjUzBRMB0GA1Ud\n" \

"DgQWBBTUkUBVumJTTBuD5v3AFzeCZawJZDAfBgNVHSMEGDAWgBTUkUBVumJTTBuD\n" \

"5v3AFzeCZawJZDAPBgNVHRMBAf8EBTADAQH/MA0GCSqGSIb3DQEBCwUAA4IBAQDQ\n" \

"lQ9GZ1f795z/uuwDEZMZdlRLny5zYKyWyAbAFvhlbuSJ7zBLpefxezJ4a+eLCRqE\n" \

"vQC1hYYor7veAsIXWaXklg0DzofV6U6z4iXdwJ3iy6aYuppvsZf1lLc56ahRPdlL\n" \

"nytxX01cinl4tqhPNABpqlOPc8/RoDORvT9kN+eoC1zfJXd+YzBpYzTuANH6mfpf\n" \

"sW4jgYn8MJNddduKtvxDqS8iYYrxRgdBa9bbGNJ2OXAytD61iO0Aw7dnHgJVD3Nw\n" \

"25U3eo14ZTCNpGaSsLtHnkOEvxgRccgrNoTh0QIr/vLcTa1+LmHGHUsnM733pC0o\n" \

"oBG4NO1ZwjEI09SisKWQ\n" \

"-----END CERTIFICATE-----\n";

const char\* key = \

"-----BEGIN PRIVATE KEY-----\n" \

"MIIEvAIBADANBgkqhkiG9w0BAQEFAASCBKYwggSiAgEAAoIBAQDiOMss0W6omn0W\n" \

"JEur6/+EgnrwmaqMCpDHgLj7xlzFHhw3PyyGPUt+UTB95w2dmQtzM+36RDuSTYa9\n" \

"w/4p4eZ8cqzyDXK3fD9tz7Db011Ln9yZYujGqcsKi6PPYlJWYk/UuV6rggyfTx4b\n" \

"1ioF/xkL+cQ2OvqinBZLbTBcojkSWTNRvnveQ4AMLghszdk62pmYTzeZeN2Cx06+\n" \

"0Kxit8PC0y/SG0rF+P0tYZiZvtMK2Cj5GCiSUBX3aRJNrpbga+9Vir2OZmxWzMeZ\n" \

"rgtQ50/Ji3YAbDZXdBo/t6D6ECsrZLbtoaTS6nz9UQRooPjUzNl1tdONVuiLkRoo\n" \

"4vsr+vu3AgMBAAECggEAFHY49mr+cd9WEh3NVGsfwl9qTv9Eka8rokf5M24hRQPG\n" \

"CenmapalEJ4NCfj6CSJnNTqMu3bW1pMO7XTgCpoawXCT8RRDUee59az0B9DUnvsC\n" \

"NpjfijH1zL84XWaSIQB6d7jJSy++WYiL+3jsddIO9L+Dk7jTo8gE149e8PoMDyXO\n" \

"eG4s5y+FweGKGKt/8zxc1InLwokBE2C8yENRVe9JgkWfM4gdmGuUUa7uuv/tqNrw\n" \

"skRAvlYJui0Gzg9ip5o5oL3YJwgH/On5YH2n6ZNGXFY9GG2nQMtpbMP0N0RPW2l1\n" \

"K5X39aqukfMrOsSh6qMdnxcLp6CthF1KcmsaD0o2jQKBgQD0Dmqlu4rdlViLkbkN\n" \

"+CYbafCMAJKnjMVYqUDNvA0zbrVRD1ATiim12gGkf2Idv8YPd9DGGXH1Qo4QsVpY\n" \

"hWrNCs6lBcZAYcqXN12qSDwCxBnFrZeSRZMfDcbZtwlNPZTy2PaYE7SOS6zKMoJG\n" \

"0NSAuY6dE72aPQFr+JjjUbu53QKBgQDtSvGLfpTOmOx6ZHAKgO4Onw74zLm+885Y\n" \

"0nB0bIZ4R1BE/XwDn+wCNZn9/tpZ67nTNBfUQagQL4z2b6UUJD/ozjOeMV4PZiaL\n" \

"3GR40cWCqSiG+QzluN9Sz+CZSbKtbbcYjt1Le7gVx2HxgM1EZ50KP+9/1WBlDpMv\n" \

"x/T3lTH0owKBgAOZ3tr5LHBejsLocImTOsS2yHTYYVGVkyfev3UJrEww+v3qMu1G\n" \

"4ZysTc/mjUURGlQKGYuCvpq206/NXh7mQRArc6cdsbIrfzjB+WDU1gXYbOTszcq7\n" \

"uhMAOgnmWH3KRdNbfi/6QnTDCxakriVrWWdY1ZDdxLNs6QN11JOuFg4FAoGAPOAm\n" \

"GYBO0rWerUOJDtjGuc+Ker+iw18ux9Cs8uFImiDJcfmJ0DA9zs9Lg0LXZ4w27v0F\n" \

"Q/nYENbyFeLMed9W2fXUfjfyBodRSj4vNGO0hPG6qO3JpkGIYWldGFY01ZM5Dr7P\n" \

"/OZvcn7lHJzLjEn8mr2ORsBJ64MStYZZzQ5kc3sCgYB6IJ2VEc0jVpsEwIZ5UCWv\n" \

"SU5+88UkhsZzic3Pl3Y0VgTDkc0vWXwOmuIKZr9/hOLAOcIKSFGrmi5fz9lHYrLz\n" \

"fPi2aAWFdPNeBfSz/UYxiB9OuPfB/2aI3+b2Y9zdDV+BGGIEgmZ4MApSulbaOYMx\n" \

"K3BgnT1nvVCA69utd/IjVg==\n" \

"-----END PRIVATE KEY-----\n";

CRGB leds[NUM\_LEDS]; // масив led-ів

BearSSL::WebServer server(443); // HTTPS для ESP32

arduinoFFT FFT = arduinoFFT(); // ініціалізація бібліотеки для обробки FFT

double vReal[SAMPLES]; // масив дійсних частин сигналу (мова про комплексні числа - зберігаються одночасно у двох масивах)

double vImag[SAMPLES]; // масив уявних частин сигналу (мова про комплексні числа - зберігаються одночасно у двох масивах)

volatile int mode = 1; // volatile, щоб уникнути проблем із перериваннями

int ampR; // накопичення суми амплітуд басових частотних діапазонів для індикації червоним кольором

int ampG; // накопичення суми амплітуд середніх частотних діапазонів для індикації зеленим кольором

int ampB; // накопичення суми амплітуд високих частотних діапазонів для індикації синім кольором

#define noiseThreshold 0 // поріг шуму, підбираємо експериментально залежно від зашумленості приміщення

const int porig = 1; // поріг для визначення кількості led-ів, які будуть заповнені кольором в залежності від накопиченого значення амплітуди

void handleMode1() {

  Serial.println("Отримано запит: Режим 1 - початок");

  mode = 1;

  server.send(200, "text/plain", "OK");

  Serial.println("Отримано запит: Режим 1 - кінець");

}

void handleMode2() {

  Serial.println("Отримано запит: Режим 2 - початок");

  mode = 2;

  server.send(200, "text/plain", "OK");

  Serial.println("Отримано запит: Режим 2 - кінець");

}

void handleMode3() {

  Serial.println("Отримано запит: Режим 3 - початок");

  mode = 3;

  server.send(200, "text/plain", "OK");

  Serial.println("Отримано запит: Режим 3 - кінець");

}

// Задача для ядра 0: Обробка HTTPS

void webServerTask(void \*pvParameters) {

  server.on("/mode1", handleMode1);

  server.on("/mode2", handleMode2);

  server.on("/mode3", handleMode3);

  server.setRSACert(new BearSSL::X509List(cert), new BearSSL::PrivateKey(key));

  server.begin();

  Serial.println("HTTPS-сервер запущено на ядрі 0!");

  for (;;) {

    server.handleClient();

    vTaskDelay(10 / portTICK\_PERIOD\_MS);

  }

}

// Задача для ядра 1: FFT і світлодіоди

void lightMusicTask(void \*pvParameters) {

  for (;;) {

    Serial.println("Починаю FFT на ядрі 1...");

    for (int i = 0; i < SAMPLES; i++) { // забір даних із мікрофона

      vReal[i] = analogRead(A0); // Зміни на свій пін (наприклад, 34 для ESP32)

      vImag[i] = 0;

      delayMicroseconds(1000000 / SAMPLING\_FREQ); // забір даних робимо з затримкою, вказаною у змінній SAMPLING\_FREQ, мкс

    }

    Serial.println("FFT закінчено");

    // Виконуємо послідовно три процедури Fast Fourier Transform, FFT:

    FFT.Windowing(vReal, SAMPLES, FFT\_WIN\_TYP\_HAMMING, FFT\_FORWARD); // Функція для зменшення впливу країв сигналу

    /\*

    Коли ми беремо скінченний набір зразків сигналу (наприклад, 128 зразків із частотою 10 кГц, як у нас), ми фактично "вирізаємо" шматок із безперервного сигналу.

    Цей процес обрізання називається truncation, і він створює проблему: краї обрізаного сигналу (початок і кінець) стають різкими перепадами (discontinuities),

    навіть якщо оригінальний сигнал був плавним (наприклад, синусоїда).

    Ці різкі перепади призводять до появи спектральних витоків (spectral leakage) у частотній області після FFT.

    Спектральні витоки — це коли енергія однієї частоти "розмазується" на сусідні частоти, спотворюючи результат аналізу спектра.

    Віконне зважування вирішує цю проблему, згладжуючи краї сигналу перед FFT:

      1) множить кожен зразок сигналу на певну вагу, яка залежить від його позиції у наборі даних.

      2) зазвичай ваги на краях (біля 0 і 127, як у нас) близькі до 0, а в центрі (біля 64) — максимальні.

    Використовуємо одну з популярних віконних функцій - вікно Хеммінга (FFT\_WIN\_TYP\_HAMMING).

    До віконування масив vReal містить "сирі" зразки з мікрофона, наприклад: [100, 102, 105, ..., 98].

    Після віконування кожен елемент vReal[i] множиться на відповідне значення вікна Хеммінга:

      vReal[0] = vReal[0] \* 0.08 (зменшується).

      vReal[63] = vReal[63] \* 1.0 (залишається майже без змін).

      vReal[127] = vReal[127] \* 0.08 (зменшується).

    Результат:

      Сигнал стає плавнішим на краях, що зменшує різкі перепади.

      Зменшується вплив країв сигналу - без віконування обрізаний сигнал виглядає як прямокутник (усі зразки мають однакову вагу),

      що додає високочастотні артефакти в спектр.

      Вікно Хеммінга "згладжує" краї, роблячи сигнал схожим на дзвін, що знижує ці артефакти.

      Хоча вікно трохи розширює основну частотну складову (main lobe), воно значно зменшує бічні піки (side lobes), що робить спектр чіткішим.

      Додається точність аналізу - це важливо для коректного розподілу частот на світлодіоди (низькі, середні, високі), щоб шум від різких країв не спотворював результат.

    Бібліотека arduinoFFT підтримує й інші вікна:

      FFT\_WIN\_TYP\_RECTANGLE (без зважування, прямокутне вікно — найгірше для витоків).

      FFT\_WIN\_TYP\_HANN (вікно Ханна — схоже на Хеммінга, але з іншими характеристиками).

      FFT\_WIN\_TYP\_BLACKMAN (ще сильніше придушення бічних піків).

      Хеммінг — хороший компроміс між роздільністю і придушенням витоків.

  FFT.Compute(vReal, vImag, SAMPLES, FFT\_FORWARD); // Виконання FFT

  FFT.ComplexToMagnitude(vReal, vImag, SAMPLES);

  // Готуємо виведення на кружальце. У трьох змінних - сумарні амплітуди частот для трьох різних кольорів

   for (int i = 0; i < SAMPLES; i++) {

    if (i < 43)  ampR += vReal[i];

    else if (i < 86) ampG += vReal[i];

    else if (i < 128) ampB += vReal[i];

    }

  // Робимо поправку на шум, значення noiseThreshold підберемо експериментально

  ampR -= noiseThreshold \* 6;   if (ampR < 0) ampR = 0;

  ampG -= noiseThreshold \* 5;   if (ampG < 0) ampG = 0;

  ampB -= noiseThreshold \* 5;   if (ampB < 0) ampB = 0;

  FastLED.clear();

  if (ampR < porig) leds[0] = CRGB(255, 0, 0); // Заповнення перших 6-ти led=ів червоним кольором залежно від накопиченого сумарного значення амплітуди басів

  else if (ampR < porig \* 2) std::fill(leds + 0, leds + 2, CRGB(255, 0, 0)); // Заповнює елементи з індексами 0, 1

  else if (ampR < porig \* 3) std::fill(leds + 0, leds + 3, CRGB(255, 0, 0)); // Заповнює елементи з індексами 0, 1, 2

  else if (ampR < porig \* 4) std::fill(leds + 0, leds + 4, CRGB(255, 0, 0)); // Заповнює елементи з індексами 0, 1, 2, 3

  else if (ampR < porig \* 5) std::fill(leds + 0, leds + 5, CRGB(255, 0, 0)); // Заповнює елементи з індексами 0, 1, 2, 3, 4

  else if (ampR < porig \* 6) std::fill(leds + 0, leds + 6, CRGB(255, 0, 0)); // Заповнює елементи з індексами 0, 1, 2, 3, 4, 5

  if (ampG < porig) leds[6] = CRGB(0, 255, 0); // Заповнення наступних 5-ти led=ів зеленим кольором залежно від накопиченого сумарного значення амплітуди середніх частот

  else if (ampG < porig \* 2) std::fill(leds + 6, leds + 8, CRGB(0, 255, 0)); // Заповнює елементи з індексами 0, 1

  else if (ampG < porig \* 3) std::fill(leds + 6, leds + 9, CRGB(0, 255, 0)); // Заповнює елементи з індексами 0, 1, 3

  else if (ampG < porig \* 4) std::fill(leds + 6, leds + 10, CRGB(0, 255, 0)); // Заповнює елементи з індексами 1, 2, 3, 4

  else if (ampG < porig \* 5) std::fill(leds + 6, leds + 11, CRGB(0, 255, 0)); // Заповнює елементи з індексами 1, 2

  if (ampB < porig) leds[11] = CRGB(0, 0, 255); // Заповнення наступних 5-ти led=ів синім кольором залежно від накопиченого сумарного значення амплітуди високих частот

  else if (ampB < porig \* 2) std::fill(leds + 11, leds + 13, CRGB(0, 0, 255)); // Заповнює елементи з індексами 0, 1

  else if (ampB < porig \* 3) std::fill(leds + 11, leds + 14, CRGB(0, 0, 255)); // Заповнює елементи з індексами 0, 1, 3

  else if (ampB < porig \* 4) std::fill(leds + 11, leds + 15, CRGB(0, 0, 255)); // Заповнює елементи з індексами 1, 2, 3, 4

  else if (ampB < porig \* 5) std::fill(leds + 11, leds + 16, CRGB(0, 0, 255)); // Заповнює елементи з індексами 1, 2

  if (mode == 1) Serial.println("Mode #1");

  else if (mode == 2) Serial.println("Mode #2");

  else if (mode == 3) Serial.println("Mode #3");

  FastLED.show();

  vTaskDelay(50 / portTICK\_PERIOD\_MS); // 50 мс затримка

  }

}

void setup() {

  Serial.begin(74880);

  pinMode(34, INPUT);

  FastLED.addLeds<WS2812B, LED\_PIN, GRB>(leds, NUM\_LEDS);

  FastLED.setBrightness(100);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL\_CONNECTED) {

    delay(500);

    Serial.print(".");

  }

  Serial.println("");

  Serial.println("Wi-Fi підключено!");

  Serial.print("IP-адреса: ");

  Serial.println(WiFi.localIP());

//  server.setServerKeyAndCert(key, cert);

  // Створюємо задачі на різних ядрах

  xTaskCreatePinnedToCore(

    webServerTask,    // Функція задачі

    "WebServerTask",  // Назва задачі

    8192,             // Розмір стеку (більше для HTTPS)

    NULL,             // Параметри

    1,                // Пріоритет

    NULL,             // Хендл задачі

    0                 // Ядро 0

  );

  xTaskCreatePinnedToCore(

    lightMusicTask,   // Функція задачі

    "LightMusicTask", // Назва задачі

    8192,             // Розмір стеку

    NULL,             // Параметри

    1,                // Пріоритет

    NULL,             // Хендл задачі

    1                 // Ядро 1

  );

}

void loop() {

}

\*/

185.199.98.49