```
// PARCIAL 5 - DISEÑO CON uP y uC. 2024-1.
// NOMBRE: Diego Andrés García Díaz.
// CÓDIGO: 2195533.
// ------------------------------------------------------------------------

// Incluye las librerías necesarias
#include <arduinoFFT.h>  // Incluimos la biblioteca de la FFT
#include <math.h>
#include <cmath>

// Declaración de variables globales
hw_timer_t *timerCore1 = NULL; // Temporizador para el núcleo 1
hw_timer_t *timerCore2 = NULL; // Temporizador para el núcleo 2

float time_read1 = 0; // Variable para el tiempo de ejecución de FFT de F1
float time_read2 = 0; // Variable para el tiempo de ejecución de FFT de F2

int Prescaler = 80; // Valor del prescaler para el temporizador

float Fs = 2048; // Frecuencia de muestreo en Hz
const int muestras = 2048; // Número de muestras
int f = 1; // Frecuencia de las señales en Hz
float w = 2 * PI * f; // Frecuencia angular

double F1[muestras]; // Arreglo para la señal F1
double F2[muestras]; // Arreglo para la señal F2

double Img1[muestras] = { 0.0 }; // Arreglo de valores imaginarios para F1 (inicializado a 0)
double Img2[muestras] = { 0.0 }; // Arreglo de valores imaginarios para F2 (inicializado a 0)

arduinoFFT FFT = arduinoFFT(); // Objeto de la clase arduinoFFT para calcular la FFT

// Función para generar las señales F1 y F2
void senales() {
  // Señal F1: Suma de señales senoidales
  for (float i = 0; i < muestras; i++) {
    int n = i;
    F1[n] = 50 + 1200 * sin(w * (i / Fs)) + 600 * sin(3 * w * (i / Fs)) + 300 * sin(5 * w * (i / Fs)) +
150 * sin(20 * w * (i / Fs));
```

```
  }
  // Señal F2: Onda cuadrada con ciclo de trabajo al 50%
  for (int i = 0; i < (muestras / 2); i++) {
    F2[i] = 1000;
    F2[i + 1024] = -1000;
  }
}

// Función para calcular la FFT de un arreglo de muestras
bool FFT_calculate(double vReal[], double Img[]) {
  FFT.Compute(vReal, Img, muestras, FFT_FORWARD); // Calculamos la FFT
  FFT.ComplexToMagnitude(vReal, Img, muestras); // Convertimos los resultados a magnitudes

  vReal[0] = vReal[0] / muestras; // Normalizamos el componente DC
  vReal[20] = 2 * vReal[20] / muestras; // Normalizamos la componente en 20 Hz

  for (int i = 1; i < muestras / 2; i = i + 2) {
    vReal[i] = 2 * vReal[i] / muestras; // Normalizamos los demás componentes
  }

  return true;
}

// Función para ejecutar la FFT de F1 en el núcleo 1
void core1Task(void *param) {
  FFT_calculate(F1, Img1); // Calculamos la FFT de F1
  timerStop(timerCore1); // Detenemos el temporizador del núcleo 1
  time_read1 = timerRead(timerCore1) / 1000.0; // Medimos el tiempo de ejecución
  vTaskDelete(NULL); // Borramos la tarea del núcleo 1
}

// Función para ejecutar la FFT de F2 en el núcleo 2
void core2Task(void *param) {
  FFT_calculate(F2, Img2); // Calculamos la FFT de F2
  timerStop(timerCore2); // Detenemos el temporizador del núcleo 2
  time_read2 = timerRead(timerCore2) / 1000.0; // Medimos el tiempo de ejecución
  vTaskDelete(NULL); // Borramos la tarea del núcleo 2
}
```

```
void setup() {
  senales(); // Generamos las señales F1 y F2
  Serial.begin(115200);

  // Configuramos los temporizadores para cada núcleo
  timerCore1 = timerBegin(1, Prescaler, true); // Temporizador para el núcleo 1
  timerCore2 = timerBegin(0, Prescaler, true); // Temporizador para el núcleo 2

  // Creamos y ejecutamos la tarea para el núcleo 1 (FFT de F1)
  xTaskCreatePinnedToCore(core1Task, "Core1Task", 10000, NULL, 1, NULL, 1);

  // Creamos y ejecutamos la tarea para el núcleo 2 (FFT de F2)
  xTaskCreatePinnedToCore(core2Task, "Core2Task", 10000, NULL, 1, NULL, 0);
}

void loop() {
  // Esperamos a que las tareas en los núcleos 1 y 2 terminen
  while (time_read1 == 0 || time_read2 == 0) {
    delay(10);
  }

  // Mostramos los resultados y tiempos de ejecución en el monitor serial
  Serial.println("Tiempo de ejecución FFT F1 (Núcleo 1): " + String(time_read1) + " [ms]");
  Serial.println("Tiempo de ejecución FFT F2 (Núcleo 2): " + String(time_read2) + " [ms]");
  Serial.println();

  // Imprime los primeros 10 armónicos de las señales F1 y F2
  for (int i = 0; i < 11; i++) {
    Serial.print(i);
    Serial.print(" Coeficiente --> F1: ");
    Serial.print(F1[i]);
    Serial.print("\t | F2: ");
    Serial.println(F2[i]);
  }

  // Detenemos el programa después de mostrar los resultados una vez
  while (1) {
  delay(1000);
  }
```

}