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

// Incluye las librerías necesarias
#include <arduinoFFT.h>  // Otra librería a usar podría ser fft.h
#include <math.h>
#include <cmath>

// Declaración de variables globales
hw_timer_t * timer = NULL; // Temporizador para el núcleo 1
float time_read = 0; // Variable para el tiempo de ejecución de la FFT
int Prescaler = 80;

float Fs = 2048;
const int muestras = 2048;
int f = 1;
float w = 2 * PI * f;

// Declaración de arreglos para almacenar las señales
double F1[muestras];
double F2[muestras];

// Arreglos para la parte imaginaria de la FFT (inicializados en 0.0)
double Img1[muestras] = { 0.0 };
double Img2[muestras] = { 0.0 };

// Creación de un objeto FFT
arduinoFFT FFT = arduinoFFT();

// 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 conjunto de datos
bool FFT_calculate(double vReal[], double Img[]) {
  // Calcula la FFT
  FFT.Compute(vReal, Img, muestras, FFT_FORWARD);
  // Convierte los resultados a magnitudes
  FFT.ComplexToMagnitude(vReal, Img, muestras);
  // Normaliza los valores de la FFT
  vReal[0] = vReal[0] / muestras;
  vReal[20] = 2 * vReal[20] / muestras;
  for (int i = 1; i < muestras / 2; i = i + 2) {
    vReal[i] = 2 * vReal[i] / muestras;
  }
  return true;
}

void setup() {
  // Genera las señales F1 y F2
  senales();
  // Inicia la comunicación serial
  Serial.begin(115200);
}

void loop() {
  // Imprime un espacio en blanco en el monitor serial
  Serial.println();

  // Inicia un temporizador para medir el tiempo de ejecución de la FFT
  timer = timerBegin(1, Prescaler, true);

  // Calcula la FFT de las señales F1 y F2
  Serial.println(FFT_calculate(F1, Img1));
  Serial.println(FFT_calculate(F2, Img2));
```

```
  // Detiene el temporizador y calcula el tiempo transcurrido
  timerStop(timer);
  float time_read = timerRead(timer) / 1000;

  // Imprime el tiempo de ejecución en milisegundos
  Serial.println();
  Serial.println("\n\n");
  Serial.print("Tiempo de ejecución FFT F1 y F2 (Núcleo 1): ");
  Serial.print(time_read);
  Serial.print(" [ms] ");
  Serial.println();
  Serial.println("\n\n");

  // 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]);
  }

  // Bucle infinito para mantener el microcontrolador en espera
  while (1)
    ;
}
```