```c
#include <stdio.h>
#include "pico/stdlib.h"
#include "hardware/adc.h"
#include "hardware/timer.h"
#include "hardware/gpio.h"
#include "arm_math.h"

// Frecuencias deseadas
#define E4 329.63
#define B3 246.94
#define G3 196
#define D3 146.83
#define A2 110
#define E2 82.41

#define MI5 659

float32_t freq_ideal;

// Definiciones para FFT
#define MAX_SAMPLES 2048
#define SAMPLING_FREQUENCY  2000
#define SAMPLING_TIME_US    500

float32_t muestras[MAX_SAMPLES];

float32_t fft_out[MAX_SAMPLES];

float32_t fft_mag[MAX_SAMPLES / 2];

volatile bool start_fft = false;

// Definiciones de GPIO

// LED que indica que la afinacion esta por debajo
#define LED_ATRASADO    19
// LED que indica que la afinacion esta OK
```

```c
#define LED_AFINADO     20
// LED que indica que la afinacion esta por arriba
#define LED_ADELANTADO  21

// Error de frecuencia
#define ERROR_GAP   5


/**
 * @brief Toma muestras periodicas con el ADC
*/
bool muestreo_callback(struct repeating_timer *t) {
  // Variable de cuenta de muestras
  static uint16_t cant_muestras = 0;
  // Tomo muestra en tension
  muestras[cant_muestras++] = adc_read();
  // Verifico si tome las 1024 muestras
  if(cant_muestras == MAX_SAMPLES) {
    // Reseteo la cantidad de muestras
    cant_muestras = 0;
    // Inicio la FFT en el main
    start_fft = true;
    // Cancelo el timer
    cancel_repeating_timer(t);
  }

  return true;
}

/**
 * @brief Programa principal
*/
int main(){

  stdio_init_all();

  adc_init();
```

```
adc_gpio_init(26);

adc_select_input(0);

gpio_init(2);
gpio_set_dir(2, false);

gpio_init(3);
gpio_set_dir(3,false);

gpio_init(4);
gpio_set_dir(4,false);

gpio_init(5);
gpio_set_dir(5,false);

gpio_init(6);
gpio_set_dir(6,false);

gpio_init(7);
gpio_set_dir(7,false);

gpio_init(LED_ATRASADO);
gpio_set_dir(LED_ATRASADO, true);

gpio_init(LED_AFINADO);
gpio_set_dir(LED_AFINADO, true);

gpio_init(LED_ADELANTADO);
gpio_set_dir(LED_ADELANTADO, true);

// Variable para hacer uso de TIMER
repeating_timer_t muestreo;
add_repeating_timer_us(SAMPLING_TIME_US, muestreo_callback, NULL, &muestreo);

// Indicador de error de FFT
```

```
arm_status status = ARM_MATH_SUCCESS;

//arm_rfft_instance_q31 rfft_instance;
arm_rfft_fast_instance_f32 rfft_instance;

//status = arm_rfft_init_q31(&rfft_instance, MAX_SAMPLES, 0, 0);
status = arm_rfft_fast_init_f32(&rfft_instance, MAX_SAMPLES);

// Verifico que se haya podido inicializar la FFT
if(status != ARM_MATH_SUCCESS) {
  printf("Fallo la inicializacion de FFT\n");
  while (1);
}

printf("FFT inicializada correctamente\n");

while(1) {

  if (start_fft == true) {

    sleep_us(500);

    printf("Muestreo terminado!\n");
    // Hago FFT
    arm_rfft_fast_f32(&rfft_instance, muestras, fft_out, 0);
    // Obtengo las magnitudes
    arm_cmplx_mag_f32(fft_out, fft_mag, MAX_SAMPLES / 2);
    printf("FFT terminada!\n");

    float32_t max_value;
    uint32_t max_index;

    // Saco componente de DC porque por defecto es el maximo
    fft_mag[0] = 0.0;
    // Obtengo el maximo
    arm_max_f32(fft_mag, MAX_SAMPLES / 2, &max_value, &max_index);
    // Frecuencia fundamental
```

```
float freq = max_index * (SAMPLING_FREQUENCY / 2.0) / (MAX_SAMPLES / 2.0);

printf("El valor de frecuencia leido es de %.2f\n", freq);
printf("Frecuencia buscada de %.2f\n", freq_ideal);
printf("\n");

// Busco la nota deseada
if(gpio_get(2) == 1){
  freq_ideal = E4;
}
else if(gpio_get(3) == 1){
  freq_ideal = B3;
}
else if(gpio_get(4) == 1){
  freq_ideal = G3;
}
else if(gpio_get(5) == 1){
  freq_ideal = D3;
}
else if(gpio_get(6) == 1){
  freq_ideal = A2;
}
else if(gpio_get(7) == 1){
  freq_ideal = E2;
}

// Si esta por debajo de la ideal, prendo el LED_ATRASADO
if(freq < (freq_ideal - ERROR_GAP)) {
  gpio_put(LED_ATRASADO, true);
  gpio_put(LED_ADELANTADO, false);
  gpio_put(LED_AFINADO, false);
}
// Si esta por arriba de la ideal, prendo el LED_ADELANTADO
else if(freq > (freq_ideal + ERROR_GAP)) {
  gpio_put(LED_ADELANTADO, true);
  gpio_put(LED_AFINADO, false);
  gpio_put(LED_ATRASADO, false);
```

```
    }
    // Sino, prendo el LED_AFINADO
    else {
      gpio_put(LED_AFINADO, true);
      gpio_put(LED_ADELANTADO, false);
      gpio_put(LED_ATRASADO, false);
    }

    // Apago flag para que no entre de nuevo al condicional
    start_fft = false;
    // Activo de nuevo el Timer para muestrear
    add_repeating_timer_us(SAMPLING_TIME_US, muestreo_callback, NULL, &muestreo);
   }
  }
  return 0;
}
```