```
void main (void){

#include <msp430g2553.h>

#define     M           50000 //contando o tempo
#define     led_do      BIT0
#define     led_re      BIT1
#define     led_mi      BIT2
#define     led_fa      BIT3
#define     led_sol     BIT4
#define     led_la      BIT5
#define     led_si      BIT6
#define     motor       BIT7


WDTCTL = WDTPW|WDTHOLD; // Desliga WDT
P1OUT = 0; // zerando saída
P1DIR |= 0xFF; // direcionando a saída de bits <0-7>, 255 em decimal

for(;;){
      if (264 <real[i] < 297){ //frequencia_do

      P1OUT |= led_do| motor; // liga o led_do e o motor
      // controle da tensão

      else if (297 <real[i] < 330 ){ //frequencia_re

            P1OUT |= led_re| motor; // liga o led_re e o motor
            // controle da tensão

                  else if (330 <real[i] < 352 ){ //frequencia_mi

                              P1OUT |= led_mi| motor; // liga o led_mi e o motor
                              // controle da tensão

                        else if (352 <real[i] < 396 ){ //frequencia_fa

                                    P1OUT |= led_fa| motor; // liga o led_fa e o motor
                                    // controle de tensão

                              else if (396 <real[i] < 440 ){ //frequencia_sol

                                          P1OUT |= led_sol| motor; // liga o led_sol e
motor
                                    //controle de tensão

                                    else if (440 <real[i] < 495 ){ //frequencia_la

                                                P1OUT |= led_la| motor // liga o led_do
e o motor
                                    // controle de tensão
}                                                       else if (495 <real[i] < 528 ){ //frequencia_si

                                                                        P1OUT |= led_si| motor // liga o
led_si e o motor
                                                                        //controle de tensão

                                                                                }
                                                                        }
                                                                }
                                                        }
                                                }
                                        }
                                }


        }//end loop infinito
}//end main
```

FUNÇÃO

```
#include "fix_fft.h"

int real[nPts];
int imag[nPts];
int sampleInterval;

void setup()

{
  Serial.begin(115200);
  delay(500);                  //give time for serial monitor to start up in Energia
  analogReadResolution(ANALOG_RESOLUTION);

  //****************** interval calculation ******************************

  int unCorrectedSampleInterval = 500000/hiFreq;
  long startTime = micros();
  for(int i = 0; i < nPts; i++){             // determine total actual time for uncorrected interval
    real[i] = analogRead(ANALOG_IN);
    delayMicroseconds(unCorrectedSampleInterval);    // unadjusted sample interval
  }
  long endTime = micros();
  int totalTime = (int)(endTime - startTime);
  int expectedTime = nPts * unCorrectedSampleInterval;
  int errorTime = totalTime - expectedTime;
  sampleInterval = unCorrectedSampleInterval - errorTime/nPts;

}

void loop()
```

```
{
  int i;
  long startTime = micros();
  for (i=0; i<nPts; i++) {                    // read ADC pin nPts times at hiFreq kHz
    real[i] = analogRead(ANALOG_IN);
    delayMicroseconds(sampleInterval);       // adjusted sample interval
  }

  for( i=0; i<nPts; i++) imag[i] = 0;        // clear imaginary array

  fix_fft(real, imag, LOG2N, 0);             // perform fft on sampled points in real[i]

  for ( i = 0; i < nPts/2; i++)              //get the power magnitude in each bin
    {
      real[i] =sqrt((long)real[i] * (long)real[i] + (long)imag[i] * (long)imag[i]);
    }

  if (DEBUG) {
    long endTime = micros();
    Serial.print   ("\nSampling time     : ");
    Serial.print   (endTime - startTime);
    Serial.println (" micro seconds");
    // find the peak
    int peakHz = 0;
    int peaki = 0;
    for (i = 1; i < nPts/2; i++) {           // bin 0 holds the summation - not peak
      if (real[i] > peakHz) {
        peakHz = real[i];
        peaki = i;
      }
    }
    peakHz = (peaki * FREQ_RESOLUTION) - FREQ_RESOLUTION/2;
    Serial.print   ("Peak frequency    : ");
    Serial.println (peakHz);
    Serial.println ("");
  }

  while(1);
}
```