PHYSICS DEPARTMENT, IIT BOMBAY

Frequency Detection

Prasanna Siddireddy, Sandeep Subramanian, Saquib Alam
November 3, 2013

1 INTRODUCTION

The project consistis of two modes. Mode 1 is when a frequency is played into a mic it will be detected by the aurdino and the detected frequency will be displayed on the terminal. The signal x(t) which is given as input into the mic will be sampled and then fourier transform takes place. Fourier transformation gives us the input signal in the frequency domain from which we can detect the frequency. Mode 2 consists of an led grid. LEDs can be glown controlled by the frequency of the input into the mic. Initially an LED would be lit and if we say A the LED to the top glows, saying E glows one on the right, Z glows one on the down and U the one on the left.

1.1 SAMPLING

In signal processing, sampling is the reduction of a continuous signal to a discrete signal. In our case the conversion of a sound wave (a continuous signal) to a sequence of samples (a discrete-time signal) is taking place. A sample refers to a value or set of values at a point in time and/or space. A sampler is a subsystem or operation that extracts samples from a continuous signal.

Let s(t) be a continuous function to be sampled, and let sampling be performed by measuring the value of the continuous function every T seconds, which is called the sampling interval. Thus, the sampled function is given by the sequence:

s(nT), for integer values of n.

The sampling frequency or sampling rate fs is defined as the number of samples obtained in one second, thus fs = 1/T.

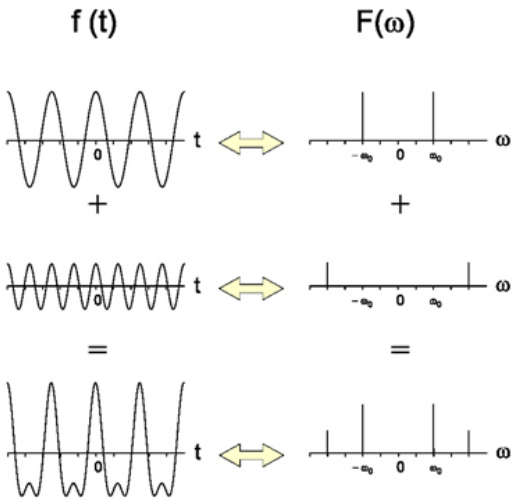

Figure 1.1: Fourier Transformation

1.1.1 Nyquist Theorem

It states that "If a function x(t) contains no frequencies higher than B hertz, it is completely determined by giving its ordinates at a series of points spaced 1/(2B) seconds apart."

$X(f) \overset{\text{def}}{=} \int_{-\infty}^{\infty} x(t)\, e^{-i2\pi f t}\, \mathrm{d}t$, and $X(f) = 0 \quad forall|f| > B$.

When x(t) is sampled uniformly at intervals of T seconds, the resultant sequence is denoted by x(nT), for all integer values of n. And the sample-rate is $f_s \overset{\text{def}}{=} 1/T$.

A sufficient condition to reconstruct x(t) from its samples is $f_s > 2B$ and equivalently $B < f_s/2$.

1.2 Fourier Transformation

It is a mathematical transformation employed to transform signals between time domain and frequency domain.

$\hat{f}(\xi) = \int_{-\infty}^{\infty} f(x)\, e^{-2\pi i x \xi}\, dx$

Here when x represents time then ξ represents frequency.

there can be the inverse transformation as follows

$f(x) = \int_{-\infty}^{\infty} \hat{f}(\xi)\, e^{2\pi i \xi x}\, d\xi$,

In our case the input is stored in an array f[], when a particular element of the array is fourier transformed it gives x the real part and y the imaginary part. Squaring and adding them gives the magnitude. Now looking for the maximum magnitude leads to the peak.

2 Circuit Connections

The power pin from the aurdino is connected to the microphone through a resistor of value 10KΩ and the other leg of the microphone is grounded. The signal from the microphone is sent to the op-amp for amplification. It is sent through a capacitor of value 2μF and resistor of value

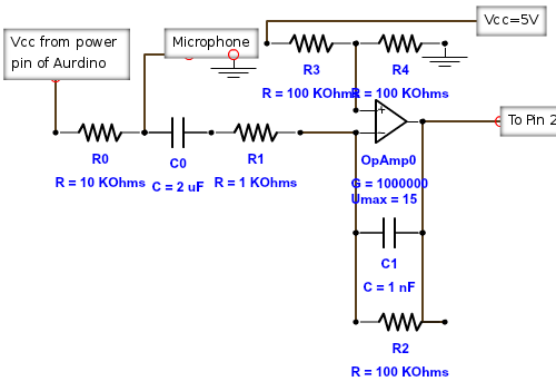

Figure 2.1: Circuit diagram

1KΩ which are in series. The op-amp is connected in the inverted amplifier configuration. The output from the op-amp goes to pin 2 of the aurdino.

3 CODE USED FOR IMPLEMENTATION

```
#define micPin 2 // MICROPHONE INPUT PIN
#define rtime 0.4 //INPUT RECORDING TIME IN SECONDS
#define twoPi 6.28318
#define samplingRate 2000 //SAMPLING RATE OF INPUT SIGNAL

byte ledx=1,ledy=1; //X AND Y CO-ORDINATES OF CURRENT GLOWING LED
int sample=0; //NO OF SAMPLES OF INPUT WAVEFORM
int trigger=0; //TRIGGER CONTROLLING THE RECORDING OF INPUT SIGNAL
byte mode=0; //STORES MODE OF OPERATION. 0 FOR FREQUENCY DETECTION
MODE, 1 FOR LED CONTROL MODE
int peakAt=0; //STORES THE PEAK FREQUENCY VALUE OF THE INPUT SPECTRA

byte f[(int)(samplingRate*rtime)-160]; //ARRAY STORING THE INPUT
SIGNAL IN TIME DOMAIN

int control[]={280,340,420,540}; //STORES MEAN FREQUENCIES OF THE
LED CONTROL ALPHABETS 'E', 'U', 'A' AND 'Z' RESPECTIVELY

void setup(){

  for(byte i=2;i<10;i++)
  pinMode(i,OUTPUT); //SETTING LED GRID PINS
```

```
  Serial.begin(9600);

  setTrigger(); //INITIALIZE TRIGGER
  Serial.println(trigger);

  Serial.println(F("Say \'A\' for frequency detection mode or \'E\'
  for LED control mode."));
  setMode();
}

void setTrigger(){ //SET TRIGGER VALUE AS AVERAGE OF MAXIMUM AND
MINIMUM NOISE VALUE OBTAINED IN 2 SECONDS

  long time=millis();
  trigger=0;
  int maxtrig=0;
  int mintrig=1024;

  while((millis()-time) < 2000)
  {
    delay(10);
    int input= analogRead(micPin);
    maxtrig = input > maxtrig? input: maxtrig;
    mintrig = input < mintrig? input: mintrig;
  }

  trigger =(maxtrig+mintrig)/2;
}

void tripTrigger(){ //EXITS OUT OF THE FUNCTION ONLY WHEN TRIGGER
CONDITION IS MET. USUALLY FOLLOWED BY START RECORD

  while(1)
  {
    if (analogRead(micPin)> trigger*1.12)
    {
      for(byte i=0;i<15;i++)
      {
        delay(10);
        if(analogRead(micPin)>trigger*1.12)
        return;
      }
    }
  }
```

```
}

void startRecord(){ //TAKES INPUT FROM micPin FOR rtime SECONDS AND
STORES IT IN f ARRAY

  sample = 0;
  long time=millis();
  int vInput=0; //TEMPORARILY STORES INPUT FROM micPIN

  while(((millis()-time) < rtime*1000))
  {
    delayMicroseconds(1000000/samplingRate);
    vInput=(analogRead(micPin)-trigger)/2+127;
    if(vInput>255)f[sample++] = 255;
    else if(vInput<0)f[sample++] = 0;
    else f[sample++] = vInput;
  }
}

/*FOURIER TRANSFORM. IF CALLED WITH ARGUMENT 0, FUNCTION SEARCHES
FOR THE BEST FREQUENCY MATCH FROM 20Hz TO MAXIMUM DETECTABLE
FREQUENCY WITH 10Hz RESOLUTION.
IF CALLED WITH ARGUMENT 1, FUNCTION SEARCHES FOR MATCHES AMONG
THE SPECIFIC FREQUENCY RANGE FOR 'A', 'E', 'U' AND 'Z'.*/

void fourierTransform(byte b){

  Serial.println(F("Recording input."));
  Serial.println(F("Processing input...."));

  peakAt=0;
  float peak=0; //STORES THE PEAK VALUE OF THE FOURIER TRANSFORM
  AMPLITUDE
  float FFT=0;  //STORES THE FOURIER TRANSFORM AMPLITUDE OF A
  PARTICULAR FREQUENCY
  float x=0,y=0; //x AND y CORRESPONDS TO THE REAL AND IMAGINARY
  PARTS OF THE FOURIER TRANSFORM

 int i=0, j=0;

  if(b==0)
  {
    for( i = 20; i < (sample/rtime)/2; i=i+10) //sample/(rtime*2)
    IS THE MAXIMUM DETECTEBLE FREQUENCY BY NYQUIST THEOREM
```

```
  {
    x = 0;
y = 0;
    for( j = 0; j< sample; j++)
    {
      x += (f[j])*cos(twoPi*i*j*rtime/sample);
      y += (f[j])*sin(twoPi*i*j*rtime/sample);
    }
    FFT = x*x + y*y;
    if(FFT > peak)
    {
      peak = FFT;
      peakAt = i;
    }
  }
}
if(b==1)
{
  for( i = 20; i < 100; i=i+10) //DETECTS NORMAL PEAK LEVEL
  {
    x = 0;
    y = 0;
    for( j = 0; j< sample; j++)
    {
      x += (f[j])*cos(twoPi*i*j*rtime/sample);
      y += (f[j])*sin(twoPi*i*j*rtime/sample);
    }
    FFT = x*x + y*y;
    if(FFT > peak)
    {
      peak = FFT;
      peakAt = i;
    }
  }
  peak = peak*5; //DONE TO ENSURE THAT THE PEAK IN THE NEXT PART
  IS VERY MUCH ABOVE NORMAL PEAK LEVEL

  for(int k=0; k<4; k++)
  {
    for( i = control[k]-15; i < control[k]+20; i=i+10)
    {
      x = 0;
      y = 0;
      for( j = 0; j< sample; j++)
```

```
        {
          x += (f[j])*cos(twoPi*i*j*rtime/sample);
          y += (f[j])*sin(twoPi*i*j*rtime/sample);
        }
        FFT = x*x + y*y;
        if(FFT > peak)
        {
          peak = FFT;
          peakAt = control[k]; //ONLY CONCERNED ABOUT THE REGION
          IN WHICH THE PEAK LIES NOT EXCAT FREQUENCY
        }
      }
    }
  }
}

void setLED(byte i, byte j) //GLOWS THE (i,j) LED IN TH XY GRID
{
  for(byte n=1;n<5;n++)
  {
    if(n!=j)
    {
      digitalWrite(n+1,HIGH);
    }
    else
    {
      digitalWrite(n+1,LOW);
    }
  }
  for(byte n=1;n<5;n++)
  {
    if(n!=i)
    {
      digitalWrite(10-n,LOW);
    }
    else
    {
      digitalWrite(10-n,HIGH);
    }
  }
}

void setMode() //RETURNS ONLY AFTER A MODE IS SET
{
```

```
  while(1)
  {
    tripTrigger();
    startRecord();
    fourierTransform(1);

    if(peakAt==control[2])
    {
      mode=0;
      Serial.println(F("Frequency detection mode selected."));
      return;
    }
    else if(peakAt==control[0])
    {
      mode=1;
      Serial.println(F("LED control mode selected."));
      Serial.println(F("Say \'A\' to go up, \'E\' to go right, \'Z\'
      to go down and \'U\' to go left"));
      setLED(ledx,ledy);
      return;
    }
    else
    {
      Serial.println(F("Bad input."));
    }
  }
}

void loop(){

  tripTrigger();
  startRecord();
  if(mode==0)
  {
    fourierTransform(0);
    Serial.print(F("The peak frequency of input signal is: "));
    Serial.println(peakAt);
  }
  if(mode==1)
  {
    fourierTransform(1);
    if(peakAt==control[0])
    {
      ledx++;
```

```
      if(ledx==5)ledx=1;
      Serial.println(F("Input \'E\'. Right."));
    }
    else if(peakAt==control[1])
    {
      ledx--;
      if(ledx==0)ledx=4;
      Serial.println(F("Input \'U\'. Left."));
    }
    else if(peakAt==control[2])
    {
      ledy++;
      if(ledy==5)ledy=1;
      Serial.println(F("Input \'A\'. Up."));
    }
    else if(peakAt==control[3])
    {
      ledy--;
      if(ledy==0)ledy=4;
      Serial.println(F("Input \'Z\'. Down."));
    }
    else
    {
      Serial.println(F("Bad input."));
    }
    setLED(ledx,ledy);
    Serial.print(F("LED co-ordinate : ("));
    Serial.print(ledx);
    Serial.print(F(","));
    Serial.print(ledy);
    Serial.println(F(")"));
  }
}
```