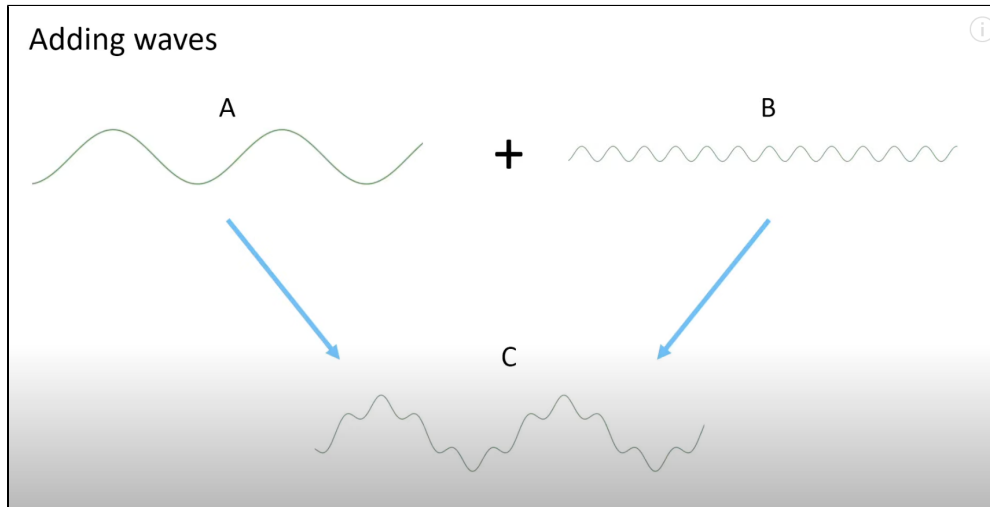


Using FFT to Read Frequencies

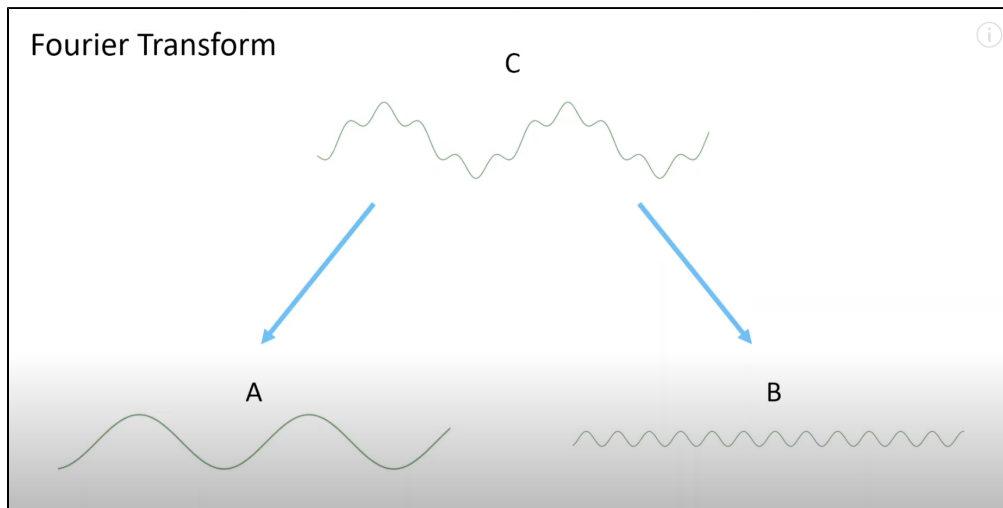
- [Background](#)
- [Sampling](#)
- [Input Circuit](#)
- [Sample Code](#)
- [References](#)

Background

We can combine waves at different frequencies and amplitudes to generate complicated waves.



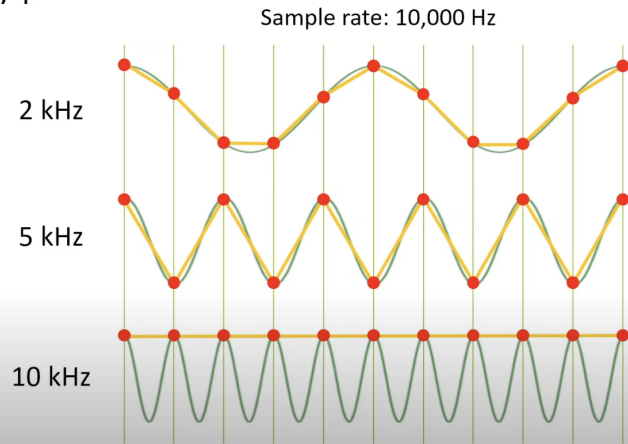
A Fourier transform takes a complicated looking wave and spits out the individual frequencies that it contains.



Sampling

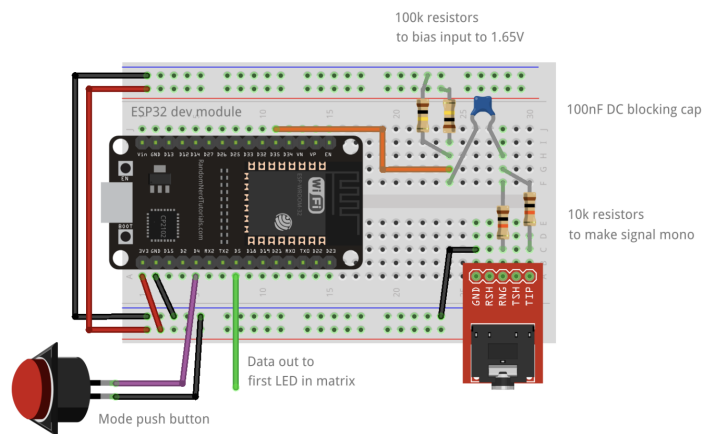
We can sample a wave with a frequency equal to half of our sampling rate. So, if we are sampling at 10,000 Hz, we can sample wave of 5000 Hz.

Nyquist theorem



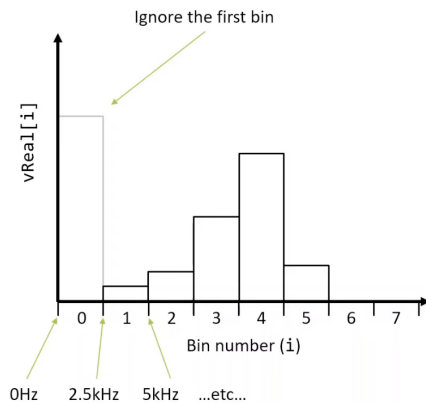
Input Circuit

Line in



Sample Code

Understanding the FFT results



Number of samples: 16
Sampling rate: 40kHz

Only use the first half of the samples

Ignore bin 0, this mainly contains DC and mains hum.

Frequency width of each bin is
sample rate / number of samples, in
this case 2.5kHz.

Size of bin = Sampling Rate/#Samples

```
#include <arduinoFFT.h>
#define AUDIO_IN_PIN    A0
#define SAMPLES         128
#define SAMPLING_FREQ   3200
#define AMPLITUDE        100
#define NUM_BANDS        8
#define NOISE            500           // Used as a crude noise filter, values below this are ignored

unsigned int sampling_period_us;

double vReal[SAMPLES];
double vImag[SAMPLES];
unsigned long newTime;

arduinoFFT FFT = arduinoFFT(vReal, vImag, SAMPLES, SAMPLING_FREQ);

void setup() {
  Serial.begin(115200);
  Serial.println("Reading...");

  sampling_period_us = round(1000000 * (1.0 / SAMPLING_FREQ));
  sampleInput();
}

void loop() {
}

void sampleInput() {

  // Sample the audio pin
  for (int i = 0; i < SAMPLES; i++) {
    newTime = micros();
    vReal[i] = analogRead(AUDIO_IN_PIN);
    vImag[i] = 0;
    while ((micros() - newTime) < sampling_period_us) { /* chill */ }
  }

  // Compute FFT
  FFT.DCRemoval();
  FFT.Windowing(FFT_WIN_TYP_HAMMING, FFT_FORWARD);
  FFT.Compute(FFT_FORWARD);
  FFT.ComplexToMagnitude();

  // binSize = SAMPLING_FREQ / SAMPLES
  // binsize = 3200/128 = 25Hz
```

```

// Analyse FFT results
int biggestFrequency = 0;
int magnitude = 0;
int binsize = SAMPLING_FREQ / SAMPLES;

// Don't use sample 0 and only first SAMPLES/2 are usable.
// Each array element represents a frequency bin and its value the amplitude.

for (int i = 1; i < (SAMPLES/2); i++){
  if (vReal[i] > NOISE) { // Add a crude noise filter
    Serial.print(i);
    Serial.print("[");
    Serial.print((i)*binsize);
    Serial.print("]");
    Serial.print(": ");
    Serial.println((int)vReal[i]);
    if((int)vReal[i] > magnitude){
      biggestFrequency=(i)*binsize;
      magnitude=(int)vReal[i];
    }
  }
}

if(biggestFrequency>0){
  Serial.print("f=");
  Serial.print(biggestFrequency);
  Serial.print(" c=");
  Serial.println(magnitude);
}
}

```

References

Reference	URL
Arduino FFT Library	https://github.com/kosme/arduinoFFT
ESP32 spectrum analyser VU meter using arduinoFFT and a FastLED matrix	https://www.youtube.com/watch?v=Mgh2WbIO5_c
ESP32_FFT_VU - Code	https://github.com/s-marley/ESP32_FFT_VU