ENGR101: Lecture 6

Signals, signal processing basics.
Project 1 programming hints.

2023

What we cover today?

- Signals - analog and digital
- Analog to digital conversion
- Project 1 introduction
- C++: better(easier) arrays
- Working with files

What is the signal?

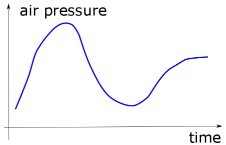
- Anything that carries information can be called a signal.
- Information in most basic form is a **measure of uncertainty removed**.
- Information is measured in bits. One bit (0 or 1, yes or no) delivered means that uncertainty was taken out of binary choice
- Any kind of **changing** physical variable can be a signal and deliver an information. Sound is a signal – changing air pressure as sensed by an ear.
  Image is a signal – changes in light intensity and colours are perceived by the eye.

Analog signal

- Analog signal is continuous - no matter how much you zoom in you will see smooth curve
- We perceive everything as an analog signals - if changes are not too fast.
- If it is too fast - we don't sense it.
- But computers do not understand analog - they work with numbers only and they do steps in time.

How to convert?

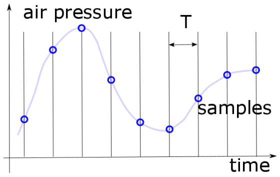
What we have...



What computer needs is a sequence of numbers: 12 23 45 56...

Analog to digital conversion, Step 1, Sampling

- First step is to measure signal at fixed moments of time.
- A sample is a value at a point in time.
- Samples are taken every T seconds
- The sampling frequency f_s or sampling rate, fs, is the number of samples obtained in one second (samples per second), thus fs $= 1/T$.
- If samples are taken at rate 10 per second ($T=0.1$ sec), for example, frequency is 10 Hz.



Step 1: Sampling rate, Nyquist theorem

- How fast samples should be taken?
- Music examples: 1024 sps(samples per second) and 44100 sps. A lot of low frequencies in 1024 sps.
- Human ear can hear acoustic signals up to 20000 Hz
- NYQUIST THEOREM:
  To detect a signal at frequency f, you must take samples at a rate of faster than 2f. To detect 1000 Hz, must digitize at $>$ 2000 samples per second (minimum).

Music: Courtesy http://music.columbia.edu/cmc/musicandcomputers/chapter2/02_03.php

Nyquist theorem

Lets have a look what happens when we sample single tone signal using different number of samples per one period of original signal.

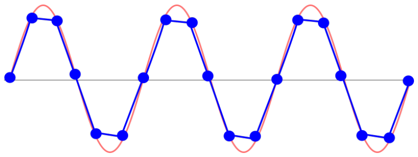


Figure: 6 samples per period

Original signal (red) can be reconstructed with high quality.

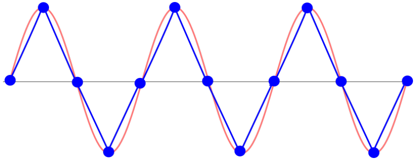


Figure: 4 samples per period

Distortions are bigger but it is still clear what original signal was.

Nyquist theorem

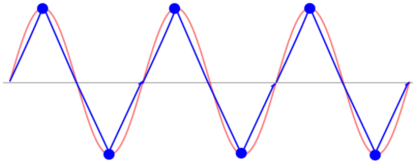


Figure: 2 samples per period - taken at right time

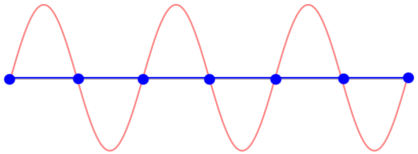


Figure: 2 samples per period - taken at wrong moments

Distortions are big, but we still can say what original signal looked like. But we had a lucky guess in this case. What if measurements are done at exactly wrong time.

We can not restore original red signal from blue measurements at all. But Nyquist theorem is sill valid: ...take samples at a rate of **faster** than 2f...

What happens if Nyquist theorem is not followed?

- Red - original signal.
- Sampling frequency is not high enough - 5 measurements for 3 periods of original signal
- Original signal can not be restored from the measurements.
- What is restored looks wrong - different frequency.

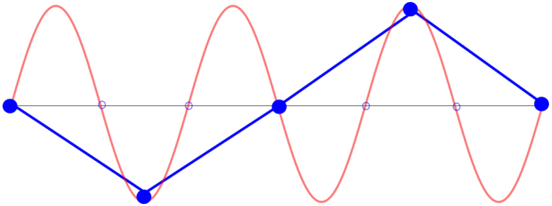


Figure: Bad sampling

Aliasing

- If time between samples is greater then half-period of original signal - samples do not represent original signal
- New frequencies will be created (aliases of original signal frequencies)
- We will hear (see) this frequency - even if it is not present in original signal - **alias**

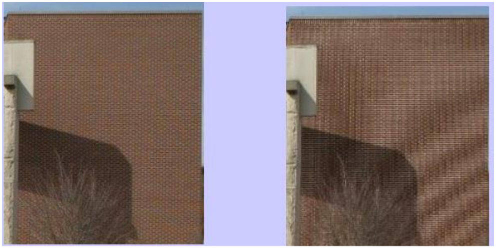
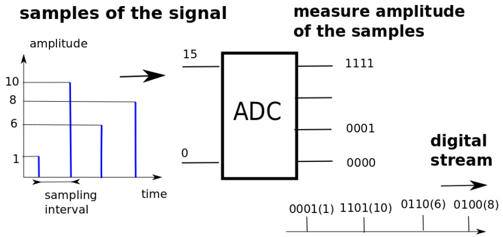


Figure: Notice false frequencies

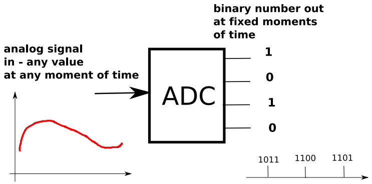
Step 2: Quantization

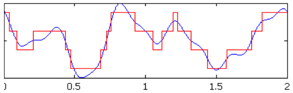


- Amplitude of each sample is measured
- Analog value is converted into number (binary)
- Fractional part of measured value is discarded

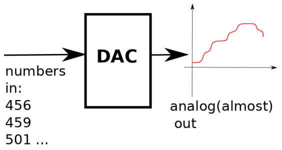
Analog to Digital Conversion (ADC)

- Process we went through is **Analog-to-Digital-Conversion**
- ADC converts continuous (analog) input signal into stream of binary numbers
- Usually it does sampling and quantization
- Output is distorted signal because digital output does not represent input signal exactly





Another way around - Digital To Analog conversion



Simple sequence of numbers (raw data) is not enough. You have to explain to DAC what is time interval between samples.
That's what Project 1 is about.

Project

- You write a software which produces digital stream (array)
- Play the files and visualize air pressure (there is a software for that)
- We made it easier for you - you don't have to follow all the bells and whistles of audio format. All you have to do is to generate the samples.
- Your software has to generate the **signal**.

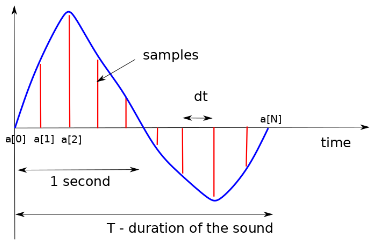
You will need:

- Make new C++ project
- **include** library functions into your project
- Make some parts of code run repeatedly (cycles)
- Make code perform different actions depending upon some condition

Reminder - what you may need to complete the project?

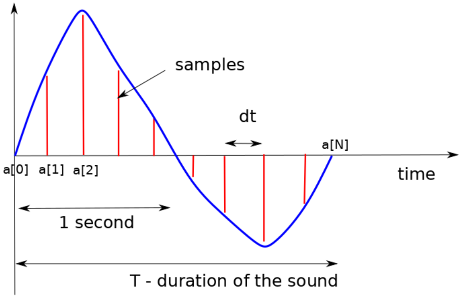
- Working with arrays
- Reserving memory for the arrays (initializing)
- Traversing (cycles)
- Working with files (challenge part)

Project



- Software should generate samples (red in picture). Hardware(speaker) will smooth it (can not move very fast) and air pressure will follow blue line (can call it envelope)
- Each sample is a number.
- Samples are stored in an array. **How big is this array?**
- First of all we need to decide what is time interval between samples (dt). Good quality sound can be produced if we use 44100 samples per second.

How big is this array?

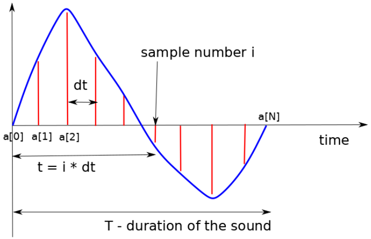


- 44100 samples per second means that samples should follow every $dt = 1/44100$ seconds
- Select duration of the sound T
- If duration is T seconds and samples should be dt seconds apart then total number of samples is:

$$N_{samples} = T/dt \qquad (1)$$

How to fill this array a[]?

We move from beginning (time=0, a[0]) to the end (time=T, a[N]).



Now we calculate amplitude (volume) of each sample:

- To calculate air pressure at time t for simple tone we use:

$$a(t) = A \cdot sin(\omega t) \qquad (2)$$

  we need to calculate t. $\omega = 2\pi f$ where f is the frequency.

- What is time of sample *i*? OK, it is sample number i, samples are *dt* apart, so it is

$$t_i = i * dt \qquad (3)$$

If *i* goes from 0 to N, t_i goes from 0 ti T.

Core (simple tone) - overall plan

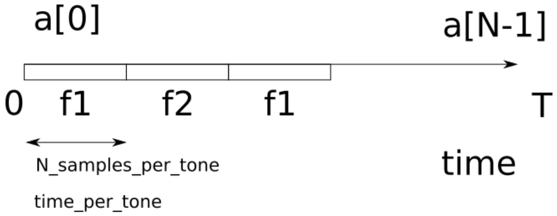
- Select variables and values: duration of the sound T, frequency f, sampling rate (44100), volume A. Choose variable types.
- Make more variables: *dt*, *N*, select their types (be careful here) and calculate values
- Run cycle for *i* going from 0 to N. At each step:
    - calculate current time
    - calculate instant air pressure *a*

$$a = A \cdot sin(\omega t), \omega = 2 \cdot \pi \cdot f \tag{4}$$

    - put *a* into array
- To use *sin* - **#include** <**math.h**>
- After whole of the array is calculated - call **MakeWavFromInt()** function (see Project script for datails). It will produce file with **wav** extension which can be played.
- If you used **vector** (details below) - use **MakeWavFromVector**.

Completion

Completion is a bit harder - frequency *f* should change at fixed moments.



Use same equation

$$a = A \cdot sin(2 \cdot \pi \cdot ft) \quad (5)$$

but *f* changes at **N_samples_per_tone** (if counting in array elements) or **time_per_tone** (if counting in seconds) since last change.

Completion

How to change the frequncy value?
There are many ways and we ask for at least two...

- Really naive one:

Listing 1: bunch of ifs

```
if ((i>=0) && (i<N_samples_per_tone)){ f=f1; }}
if ((i>=N_samples_per_tone)&&(i<2*N_samples_per_tone)){f=f2;}}..
```

- Have a look at how i/N_samples_per_tone is going
- Have a look at **modulo division**

In any case, if you have function which returns f and takes i and N_samples_per_tone as an arguments - you can test this function separately.

Challenge: Working with files

Challenge requires reading file with notes.

```
#include <iostream>
#include <fstream>
using namespace std;
int main(){
  string line;
  ifstream myfile;
  myfile.open("file.txt");
  while(getline(myfile, line)){
    cout<<"Read:"<<line<<endl;
  }
  myfile.close();
}
```

- We used **iostream** - input/output from keyboard/display. **fstream** class to perform input/output of characters from file
- You open the file first. Close after finished with it.
- **getline()** returns **true** if line was read, **false** otherwise
- **line** is modified inside the function (value, reference?)

Challenge: String is an array of characters, not a single number

```
#include <string>
#include <iostream>

int main(){
  std::string str = "345";
  int d;
  d = str; //WRONG
  return 0;
}
```

- We are reading **string** - array of characters.
- We need to convert string to int, double...
- Have a look at stod(string to double), stoi(string to int) C++ functions

If you really hate arrays... you are in good company

Advantage of C++ arrays - speed.

Disadvantage - hard to work with: once array is created - size of it can not be changed.

There is a nicer version : **vector**.

Listing 2: vector

```
#include <vector>
#include <iostream>
int main(){
  std::vector<int> my_vector;
  my_vector.push_back(34);
  my_vector.push_back(324);
  my_vector.push_back(4);
  my_vector.push_back(3);
  for (int i=0; i<my_vector.size(); i++){
   std::cout<<my_vector[i]<<" "<<my_vector.at(i)<<std::endl;
  }
  return 0;
}
```

To use vector:

- put #include <vector> before the code. It **include**s functions and types defined in **vector** class available to use in your program
- **std::vector<int> my_vector;** - declare new vector which is made out of **int**s. Name of this vector is **my_vector**. It is empty when created.
- **my_vector.push_back(34);** - add one more element to the vector. 34 is value of this element. Size of vector is adjusted automatically.
- to get i_{th} element of the vector use either **my_vector[i]** (does not check that i is valid) or **my_vector.at(i)** (does check).
- **my_vector.size()** returns current size of the vector
- **my_vector.pop_back()** deletes last element of the vector

Questions?