

pyrecplay_modulationblock

January 27, 2017

0.1 PyAudio Example:

Make a modulation between input and output (i.e., record a few samples, modulate them with a sine, and play them back immediately). Using block-wise processing instead of a for loop Gerald Schuller, October 2014.

Importing modules and defining the variables.

```
In [1]: import pyaudio
import struct
import math
import array
import numpy
import scipy

CHUNK = 5000 #Blocksize
WIDTH = 2 #2 bytes per sample
CHANNELS = 1 #2
RATE = 32000 #Sampling Rate in Hz
RECORD_SECONDS = 8
```

Initialize the sound card

```
In [2]: p = pyaudio.PyAudio()

stream = p.open(format=p.get_format_from_width(WIDTH),
                channels=CHANNELS,
                rate=RATE,
                input=True,
                output=True,
                #input_device_index=10,
                frames_per_buffer=CHUNK)
```

Start recording and playback the modulated version of it.

```
In [3]: print("* recording")

#Loop for the blocks:
for i in range(0, int(RATE / CHUNK * RECORD_SECONDS)):
```

```

    #Reading from audio input stream into data with block length "CHUNK":
    data = stream.read(CHUNK)
    #Convert from stream of bytes to a list of short integers (2 bytes here) in "samples"
    #shorts = (struct.unpack( "128h", data ))
    shorts = (struct.unpack( 'h' * CHUNK, data ));
    samples=list(shorts);

    #start block-wise signal processing:

    #Compute a block/an array of sine samples:
    s=scipy.sin(scipy.pi/128*4*scipy.arange(0,CHUNK));
    #multiply/modulate the signal with the sine samples:
    samples=samples*s;

    #end signal processing

    #converting from short integers to a stream of bytes in "data":
    data=struct.pack('h' * len(samples), *samples);
    #Writing data back to audio output stream:
    stream.write(data, CHUNK)

print("* done")

stream.stop_stream()
stream.close()

p.terminate()

* recording

c:\python27\lib\site-packages\ipykernel\__main__.py:22: DeprecationWarning: integer argument exp

* done

```