```
#include "StringSound.hpp"
#include <vector>


StringSound:: StringSound(double frequency):
  buff(ceil(SAMPLING_RATE / frequency)) {
  num = ceil(SAMPLING_RATE / frequency);

  for (int i = 0; i < num; i++) {
    buff.enqueue((int16_t)0);
  }
  tictic = 0;
}


StringSound:: StringSound(std::vector<sf::Int16> init):
  buff(init.size()) {
  num = init.size();

  std::vector<sf::Int16>::iterator it;

  for (it = init.begin(); it < init.end(); it++) {
    buff.enqueue((int16_t)*it);
  }
  tictic = 0;
}

void StringSound::pluck() {
  for (int i = 0; i < num; i++) {
    buff.dequeue();
  }

  for (int i = 0; i < num; i++) {
    buff.enqueue((sf::Int16)(rand() & 0xffff));
  }

  return;
}


void StringSound::tic() {
  int16_t first = buff.dequeue();
  int16_t second = buff.peek();

  int16_t avg = (first + second) / 2;
  int16_t karplus = avg * ENERGY_DECAY_FACTOR;

  buff.enqueue((sf::Int16)karplus);

  tictic++;

  return;
}


// return  current sample
sf::Int16 StringSound::sample() {

  sf::Int16 sample = (sf::Int16)buff.peek();

  return sample;
}


// number of tics called
```

```
int StringSound::time() {
  return tictic;
}
```