The Puget Sound, As Sound

Sonifying the tides with the Teensy Audio Adapter

Remington Furman

Wednesday, May 30th, 2018

Outline

Sonifying the tides with the Teensy Audio Adapter

- This project builds a tides simulation for use as a custom synthesizer on the Teensy 3.2 development board.
- The synthesizer uses data from NOAA to simulate the tides at a rate much faster than real time.
- When simulated fast enough, the motion of the tides can be used to drive a speaker and generate audio.

Overview

I will talk briefly about the following topics:

- What is sonification?
- How do the tides behave?
- What hardware did I use?
- What software did I write?
- What does this all sound like?

Why?

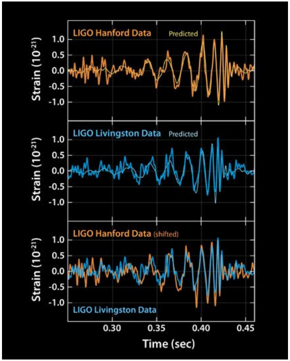
- It's fun.
- It demonstrates a great hardware and software platform for hobby projects.
- Science! Who doesn't like modeling nature with math?
- There is more than one way to present and interpret data.

Sonification

- Uses audio to present data.
- Visualization uses light to present data.
- Hearing is a very highly developed sense which our brain interprets quickly.

Sonification example

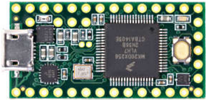
- Recent LIGO measurement of two black holes colliding. The data represents the gravitational waves that reached the Earth 1.2 billion years after the event.
- None of us have gravitational wave ears in our biology, but with a bit of software we can experience the data with our sound wave ears.
- More fun than looking at a squiggle on a graph.



How can we sonify data?

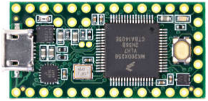
With computers and software.

Teensy 3.2



- ARM Cortex M4 microcontroller board from Portland, OR.
- Arduino on steroids.
- About $20.

Teensy 3.2



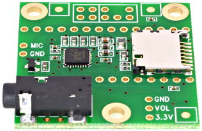
- NXP Kinetis series microcontroller
- Stats:

Clock speed:	72 MHz
Flash (program data space):	256 kB
RAM (CPU Memory):	64 kB

- Hardware floating point unit (math!)
- Tons of peripherals
  - One 12-bit DAC (generates analog signals)
  - Two 13-bit ADCs (measures analog signals)

Teensy Audio adapter

CD quality stereo sound for the Teensy.



- Stereo headphone output
- Stereo line-in input
- Mono microphone option
- CD quality: 16 bit, 44.1 kHz DAC and ADC
- Plugs right into a Teensy 3.2 board

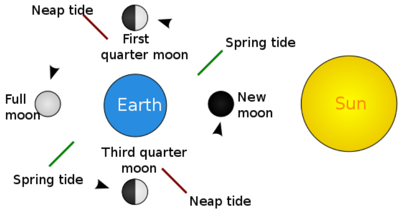
Back to the science

What are the tides?

Motion of liquid water on Earth that arises from the Earth's rotation and the gravitational pulls from the Sun and Moon.

Orbital bodies

- The tides come from the rotation and orbits of the Sun, Earth, and Moon.



- The shape of the Earth (topography) also has an effect.

Simulation math

- Orbits and rotations are periodic events. They happen in regular, measureable, and predictable cycles.
- We have math for that!

Thanks Fourier!

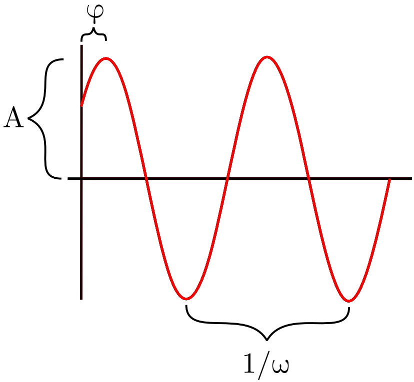
Simulation math

- Each component of the tides can be represented with a single sine wave.
- Add up all the components (sine waves) and we have a tide simulator.

Sine waves

A sine wave can be described by three properties:

- A Amplitude
- ω Frequency
- φ Phase

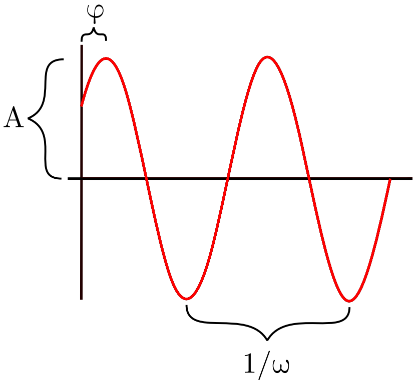


Sine waves

"Scary" trig math:

$$y = Asin(\omega t + \varphi)$$

- A Amplitude
- ω Frequency
- φ Phase
- t Time



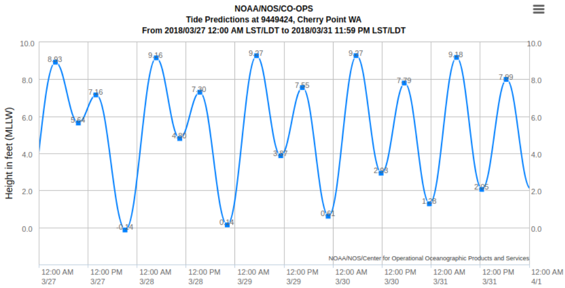
Sine waves

"Scary" trig math multiplied:

$$\begin{aligned} \textbf{tides} = {} & A_1 sin(\omega_1 t + \varphi_1) \\ & + A_2 sin(\omega_2 t + \varphi_2) \\ & + A_3 sin(\omega_3 t + \varphi_3) \\ & + A_4 sin(\omega_4 t + \varphi_4) \\ & + A_5 sin(\omega_5 t + \varphi_5) \\ & + \ldots \\ & + A_{37} sin(\omega_{37} t + \varphi_{37}) \end{aligned}$$

- Where we will get the data to plug into this equation?

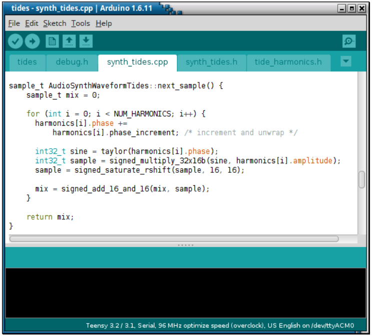
Thanks NOAA!



Const. #	Name	Amp. (m)	Phase (deg)	Frequency (deg/hr)	Description
1	M2	0.724	150.5	28.984104	Principal lunar semidiurnal constituent
2	S2	0.178	170.2	30.0	Principal solar semidiurnal constituent
3	N2	0.152	127.1	28.43973	Larger lunar elliptic semidiurnal constituent
...	...	...	...	...	...
37	MS4	0.003	52.1	58.984104	Shallow water quarter diurnal constituent

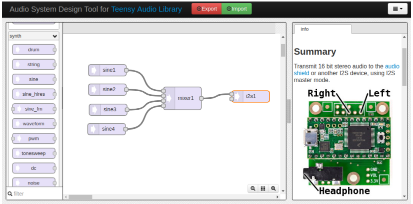
Teensy software development

- Based on Arduino IDE (super easy)
- Install the Teensy plugins and audio library
- Great software libraries for Teensy



Audio library

- Teensy Audio System Design Tool
- Easy drag and drop setup for existing audio functions.
- GUI tool automatically generates source code
- Code creates and connects C++ objects



Custom synth code

- Create a table for the data:

```
typedef struct tide_harmonic {
  float amplitude; /* feet */
  float phase;     /* degrees */
  float angular_velocity; /* degrees per hour */
  const char* name;
} tide_harmonic_t;

/* Tides table */
static const tide_harmonic_t harmonics_data[] = {
  {  3.52,  138.7,   28.984104,  "M2" },   // Principal lunar semidiurnal constituent
  {  0.88,  157.0,   30.0,       "S2" },   // Principal solar semidiurnal constituent
  {  0.71,  113.2,   28.43973,   "N2" },   // Larger lunar elliptic semidiurnal constituent
  {  2.73,  156.6,   15.041069,  "K1" },   // Lunar diurnal constituent
  {  0.07,   96.4,   57.96821,   "M4" },   // Shallow water overtides of principal lunar cons
  {  1.51,  143.0,   13.943035,  "O1" },   // Lunar diurnal constituent
  ...
  {  0.04,  118.0,   58.984104,  "MS4" },  // Shallow water quarter diurnal constituent
};
```

Custom synth code

- Read data from the table:

```
for (i = 0; i < NUM_HARMONICS; i++) {
  harmonics[i].phase =
    (M_PI * harmonics_data[i].phase) / 180.0; /* rad */

  harmonics[i].amplitude = harmonics_data[i].amplitude; /* feet, convert
                                                            at end. */

  harmonics[i].angular_velocity =
    ((harmonics_data[i].angular_velocity * DPH_2_RPS) / SAMPLE_RATE) *
                                             FREQUENCY_SCALE; /* rad/sample */
}
```

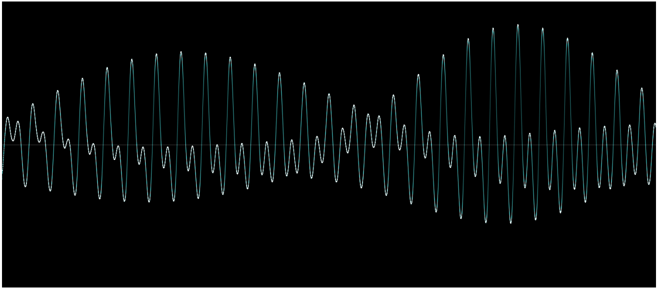
Custom synth code

- Loop through all sine waves and add them together:

```
sample_t AudioSynthWaveformTides::next_sample() {
    sample_t mix = 0;

    for (int i = 0; i < NUM_HARMONICS; i++) {
      harmonics[i].phase +=
          harmonics[i].phase_increment; /* increment and unwrap */

      int32_t sine = taylor(harmonics[i].phase);
      int32_t sample = signed_multiply_32x16b(sine, harmonics[i].amplitude);
      sample = signed_saturate_rshift(sample, 16, 16);

      mix = signed_add_16_and_16(mix, sample);
    }

    return mix;
}
```

What's the end result?

- After all this, we get an eerie tone and a pretty graph.



Where to go next?

- Use a tide simulation to:
    - Slowly amplitude modulate a tone
    - Slowly frequency modulate a tone
- Make it more interactive:
    - Add knobs to change the simulation speed and tones in real-time
    - Change location datasets with the push of a button
- Play data from a solar system orbit simulation?
- Start a Poseidon themed synth band?

Any Questions?

- Want the source code?
- Want a board?