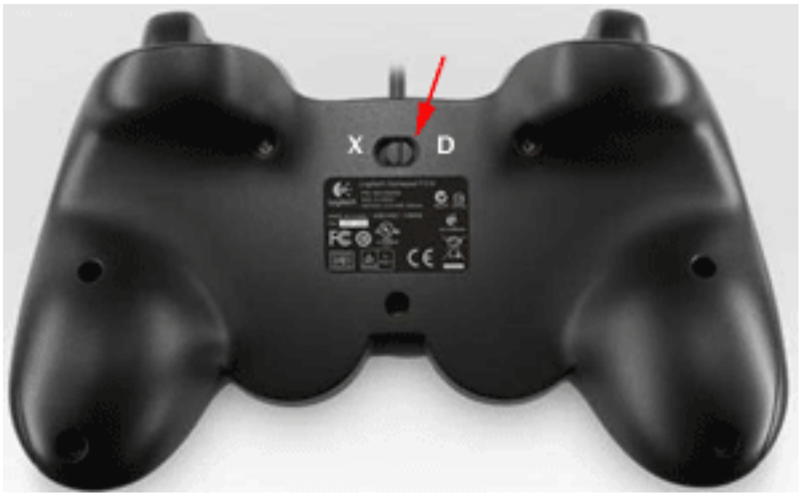
Lab 13 Controllers and Modulation

Gamepad USB Controller

Connect the Gamepad USB cable to the computer. Make sure the switch on the bottom of the gamepad is set to the D position.

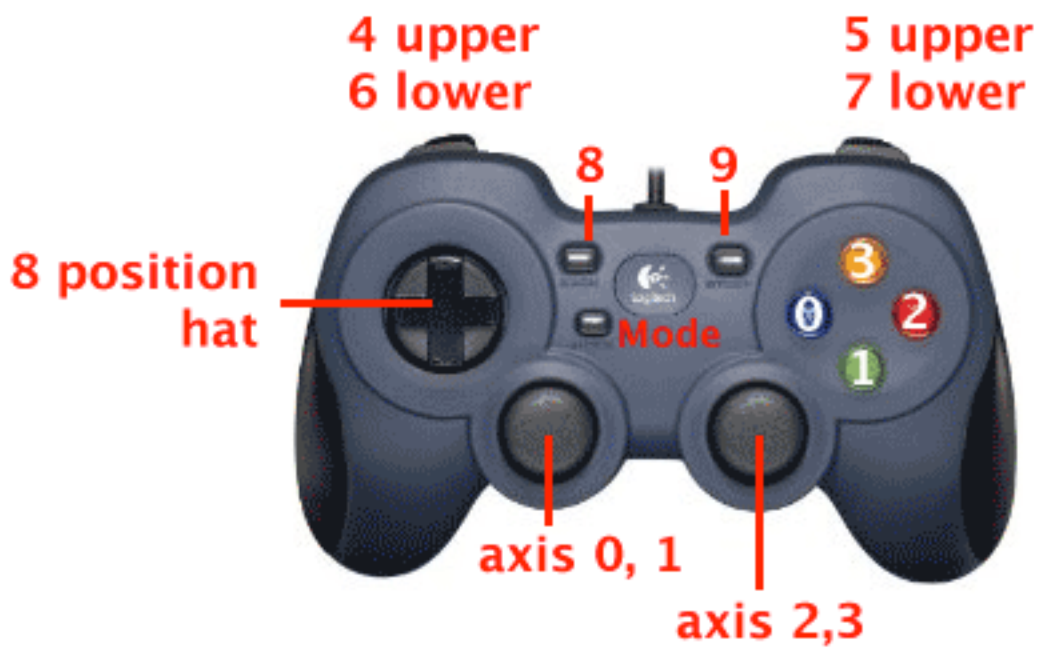


Run joy.ck

It's in the m208Lab13 folder.

Button Mapping

Operate every button and joystick on the gamepad. You should see a messages appear in the Console Monitor for every action. The button numbers should be mapped as shown.



Gampad	Values
0	down, up
1	down, up
2	down, up
3	down, up
4	down, up
5	down, up
6	down, up
7	down, up
8	down, up
axis 0,2	±1.0 east west
axis 1,3	±1.0 north south
hat	0 - 7 clockwise rotation starting from North 8 is button up
Mode	Switch hat with axis 0,1

Modulation

Modulation is the process of varying one or more properties of a periodic waveform, called the carrier, with a another waveform called the modulator. The three key parameters of a periodic waveform are its amplitude, frequency, and duration. Any of these properties can be modulated by another waveform to obtain the modulated modulated. A musician modulates a tone (a periodic waveform) on a musical instrument through touch and pressure.

The Modulator and Carrier

Modulation requires two waveforms called the modulating wave and the carrier wave. The carrier wave is modulated (modified) by the modulator wave. The modulating wave can be either bipolar or unipolar. Typically the carrier wave is the higher frequency. Modulation is sample by sample multiplication.

Ring Modulation

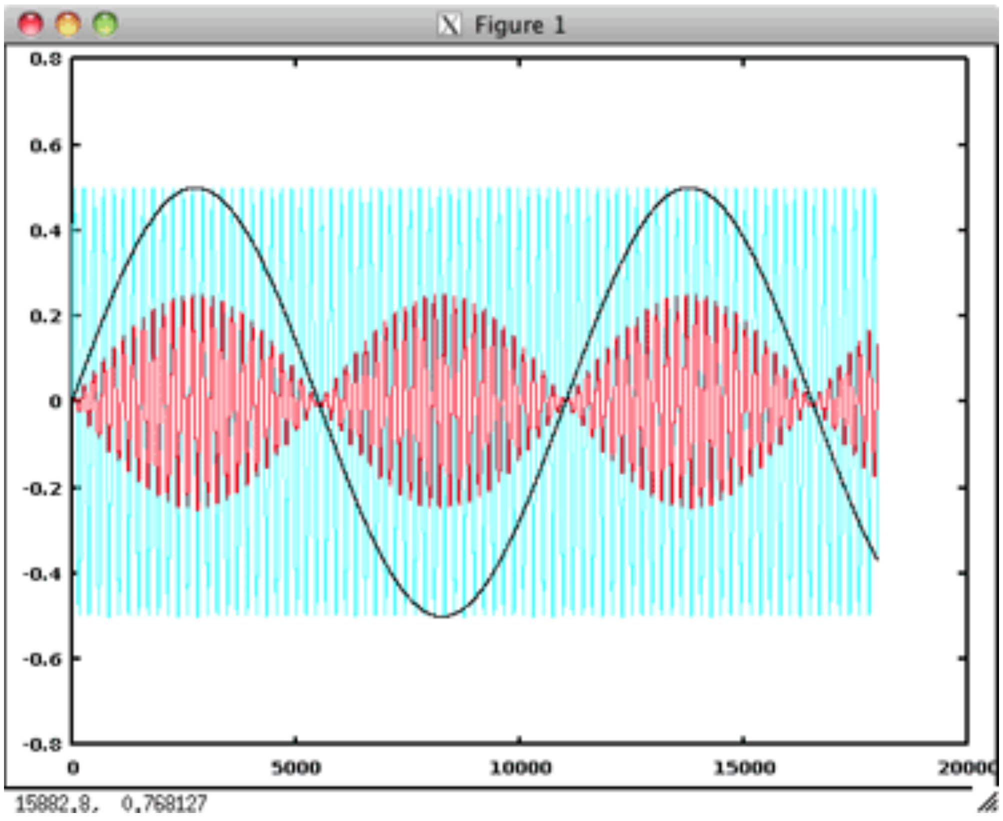
Ring Modulation is the simplest form of modulation. It multiplies two bipolar signals sample by sample. If the two signals are sinusoidal, the output of the ring modulator would be the sum and difference of the two frequencies. The fundamental frequencies of the two sinusoids will disappear. This corresponds to the trigonometric identity for multiplying two sine or cosine waves.

$$\text{Cos}(A)\,\text{Cos}(B) = \frac{\text{Cos}(A+B) + \text{Cos}(A-B)}{2}$$

and

$$\text{Sin}(A)\,\text{Sin}(B) = -\frac{\text{Cos}(A+B) - \text{Cos}(A-B)}{2}$$

In this plot of ring modulation, the modulator wave is black, the carrier wave is light blue, and the result is red.



One use of ring modulation was to create robotic voices in early sci-fi movies by ring modulating a human voice with a sine wave. Try this example.

```
// ringMod1.ck
//-------------------------------------------
// simple ring modulation - from ChucK examples
// adapted for MUSC 208 by John Ellinger
// ---
// any ugen has .op:
// -1 pass through the input
// 0 stop input
// 1 add inputs (default)
// 2 subtract from first input
// 3 multiply inputs
// 4 divide from first input
//-------------------------------------------

SndBuf carrier => Gain g => dac;
me.sourceDir() + "/snds/music208.wav" => carrier.read;
```

```
SinOsc modulator => g;
800 => modulator.freq;

// the Gain unit generator multiplies the two signals
3 => g.op;

while (true)
{
    carrier.length() => now;
    0 => carrier.pos;
}
```

Gamepad Controller and Ring Modulation

Let's try controlling the ring modulator frequency with the Gamepad controller. Enter this code. Move the right joystick to change the modulator frequency and press button 1 to stop.

```
// gamepadRingMod1.ck
// John Ellinger Music 208 Spring2014

SndBuf carrier => Gain g => dac;
me.sourceDir() + "/snds/music208.wav" => carrier.read;

SinOsc modulator => g;
0 => modulator.freq;

1 => int keepGoing;

// make HidIn and HidMsg
Hid hi;
HidMsg msg;

// which joystick
0 => int device;
// get from command line
if( me.args() ) me.arg(0) => Std.atoi => device;

// open joystick 0, exit on fail
if( !hi.openJoystick( device )  ) me.exit();

<<< "joystick '" + hi.name() + "' ready", "" >>>;
```

```
function float scaleData( float inValue, float inMin, float inMax,
float outMin, float outMax )
{
    inMax - inMin  => float inRange;
    outMax - outMin => float outRange;
    return  (outRange * inValue / inRange) + outMin;
}

function void listenForGamepad()
{
    // infinite event loop
    while( true )
    {
        // wait on HidIn as event
        hi => now;

        // messages received
        while( hi.recv( msg ) )
        {
            // joystick axis motion
            if( msg.isAxisMotion() )
            {
                if ( msg.which  == 3 ) // right north south
                {
                    1.0 +=> msg.axisPosition; // now 0 - 2.0
                    scaleData( msg.axisPosition, 0, 2.0, 20, 2000 )
=> modulator.freq;
                    <<< msg.axisPosition, modulator.freq() >>>;
                }
            }

            // joystick button down
            else if( msg.isButtonDown() )
            {
                if ( msg.which == 0 )
                {
                    0 => keepGoing;
                    <<< "joystick button", msg.which, "down" >>>;
                }
            }
        }
    }
}
```

```
function void ringMod()
{
    // the Gain unit generator multiplies the two signals
    3 => g.op;
    while (keepGoing)
    {
        carrier.length() => now;
        0 => carrier.pos;
    }
}

spork ~ listenForGamepad();
spork ~ ringMod();

3::minute => now;
```

gamepadRingMod2.ck

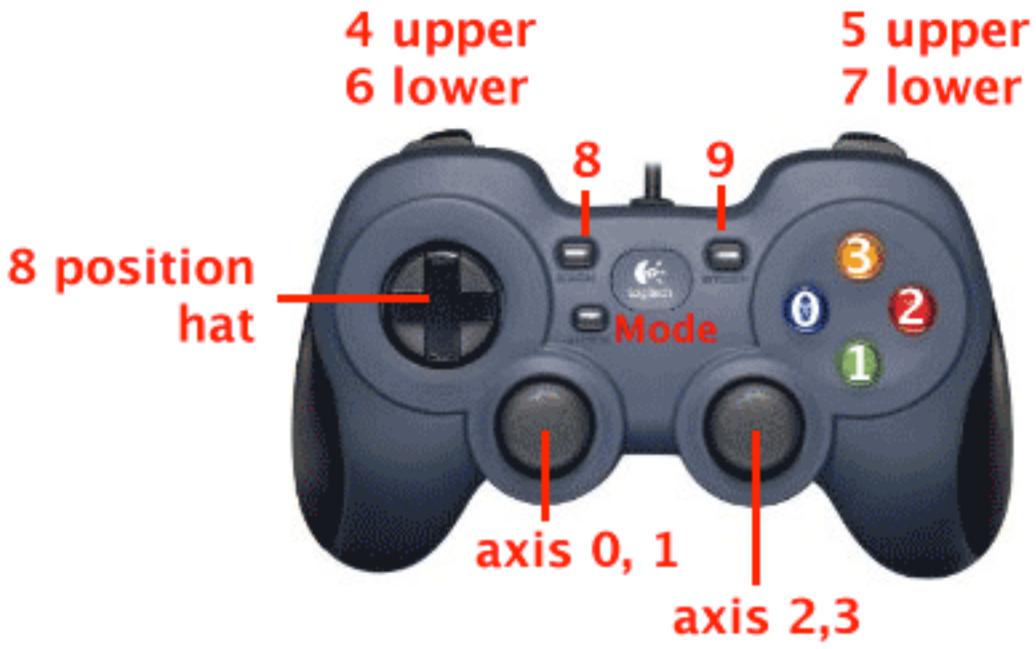
Add more gamepad controls.

```
// gamepadRingMod2.ck
// John Ellinger Music 208 Winter2014

// make HidIn and HidMsg
Hid hi;
HidMsg msg;

// which joystick
0 => int device;
// get from command line
if( me.args() ) me.arg(0) => Std.atoi => device;

// open joystick 0, exit on fail
if( !hi.openJoystick( device ) ) me.exit();

<<< "joystick '" + hi.name() + "' ready", "" >>>;

SndBuf carrier => Gain g => dac;
SndBuf modulator; // don't connect yet
SinOsc s;
3 => g.op;
3600 => s.freq;
```

```
me.sourceDir() + "/snds/music208.wav" => carrier.read;
1 => carrier.loop;
1.0 => carrier.rate;

me.sourceDir() + "/snds/chirp.wav" => modulator.read;
1 => modulator.loop;
1.0 => modulator.rate;
modulator.samples() => modulator.pos;

function float scaleData( float val, float inMin, float inMax, float
outMin, float outMax )
{
    inMax - inMin + 1 => float inRange;
    outMax - outMin => float outRange;
    return  outMin + (outRange * val / inRange);
}

// infinite event loop
while( true )
{
    // wait on HidIn as event
    hi => now;

    // messages received
    while( hi.recv( msg ) )
    {
        // joystick axis motion
        if( msg.isAxisMotion() )
        {
            if ( msg.which == 0 ) // left east west
            {
                1.0 +=> msg.axisPosition;
                scaleData(msg.axisPosition, 0, 2.0 ,400,4000) =>
s.freq;
                <<< "left axis 0 freq", msg.which, ":",
msg.axisPosition, s.freq() >>>;
            }
            else if ( msg.which == 2 ) // right east west
            {
                msg.axisPosition * 10.0 => carrier.rate;
                <<< "right axis 2 rate", msg.which, ":",
msg.axisPosition >>>;
            }
```

```
        }

        // joystick button down
        else if( msg.isButtonDown() )
        {
            if ( msg.which == 0 )
            {
                1.0 => carrier.rate;
                <<< "button 1 down", msg.which, "rate 1.0" >>>;
            }

            if ( msg.which == 4 )
            {
                0 => modulator.pos;
                modulator => g;
                <<< "button 4 down", msg.which, "connect modulator"
>>>;
            }
            else if ( msg.which == 5 )
            {
                s => g;
                <<< "button 5 down", msg.which, "add SinOsc" >>>;
            }
            else if ( msg.which == 6 )
            {
                carrier.rate() * -1.0 => carrier.rate;
                <<< "button 6 down", msg.which, "backwards" >>>;
            }
            else if ( msg.which == 7 )
            {
                Math.random2( 0, carrier.samples() ) => carrier.pos;
                <<< "button 7 down", msg.which, "random position"
>>>;
            }

        }

        // joystick button up
        else if( msg.isButtonUp() )
        {
            <<< "joystick button", msg.which, "up" >>>;
            if ( msg.which == 4 )
            {
                modulator.samples() => modulator.pos;
                modulator =< g;
                <<< "button 4 up", msg.which, "remove modulator" >>>;
            }
            else if ( msg.which == 5 )
            {
                s =< g;
                <<< "button 5 up", msg.which, "remove SinOsc" >>>;

            }
        }

        // joystick hat/POV switch/d-pad motion
        else if( msg.isHatMotion() )
        {
            <<< "joystick hat", msg.which, ":", msg.idata >>>;

        }
    }
}
```

Control Mapping



joystickLeft axis 0 (east west) SinOsc frequency (button 5 down)

joystickRight joystick axis 2 (east west) carrier playback rate change

button 1 down carrier playback rate normal 1.0

button 4 down connect modulator

button 4 up remove modulator

button 5 down connect SinOsc

button 5 up remove SinOsc

button 6 down play carrier backwards

button 7 down carrier start at random position

MUSC 208 Winter 2014
John Ellinger Carleton College