Interfacing with the Encoder and Soundboard

PyControl with PyControl Rotary Encoder

Hooked to the digital inputs on the PyControl breakout board and then connected to the PyControl board **Port 1**. Order of pins is (**B, +5V, A, I, G**) from left to right, on the side labeled "Rotary Encoder". Called in PyControl **task** program with the following code:

```
#joystick = Position_encoder_2_threshold('joystick', 100,
       #threshold_A= v.push_thresh,  rising_event_A='joystick_push', falling_event_A='push_return',
       #threshold_B= v.pull_thresh,  rising_event_B='pull_return'  , falling_event_B='joystick_pull',
       #I_pin='X12', I_pin_event='I_pin')
```

Testing: Index position drifts roughly 30-50 ticks. Small rapid movements of the joystick can lead to high (>50 tick) offset in the index position. Full tick range: [-150,150]

PyControl with NI Board (USB 6211)

NI Board is pinned to X19 on the PyControl board and the NI board must output analog signal on the range of 0-3.3 V. Called in PyControl **task** program by the following code:

```
joystick = Analog_input_2_threshold('X19', 'joystick', 100,
       threshold_A= v.push_thresh,  rising_event_A='joystick_push', falling_event_A='push_return',
       threshold_B= v.pull_thresh,  rising_event_B='pull_return', falling_event_B='joystick_pull')
```

Testing: Pending.

MATLAB Code for NI Board Readings

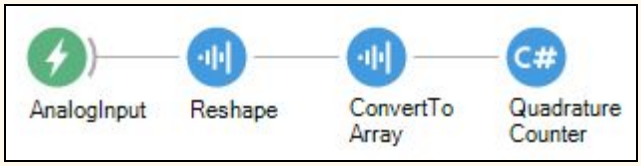
Pin the rotary encoder A and B channels into **separate** digital inputs (P0.0 and P0.1) on the NI board. In DAQ Express Test Panels, ensure your board corresponds to the following setup or edit as needed based on your setup.

```
device = 'Dev1'; % This needs to be specified for your setup
ctr = 'ctr0'; % This needs to be specified for you setup
```

Testing: MATLAB position output drifts roughly 1-2 ticks. Full tick range: [-60,60].

Bonsai directly interfacing with NI Board

Pin the A and B channels of the Rotary Encoder into separate analog inputs (AI1 and AI2, + ends). Call the encoder input with the following function in Bonsai:



AnalogInput

Ensure, in the tab "Channels" under properties, that the two analog inputs from the device that you are reading from are selected. Change SamplingRate and other parameters as needed for your experiment.

Reshape

Set "Channels" to 0 and "Rows" to 1.

ConvertToArray

Set "Depth" to F32.

CSharpTransform (QuadratureCounter.cs)

This node calls a C# definition. Double-click the transform node to create a new .cs file. Save as QuadratureCounter.cs. Open the .cs file is an editor such as Visual Studio and paste the following code:

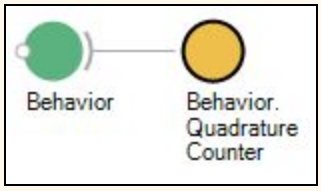
```
using Bonsai;
using System;
using System.ComponentModel;
using System.Collections.Generic;
using System.Linq;
using System.Reactive.Linq;

[Combinator]
[Description("Decodes the current state of a quadrature encoder from raw A/B channels.")]
[WorkflowElementCategory(ElementCategory.Transform)]

public class QuadratureCounter
{
	public QuadratureCounter()
	{
	Threshold = 2.5f;
	}
	public float Threshold { get; set; }
	public IObservable<int> Process(IObservable<float[]> source)
	{
	return Observable.Defer(() =>
	{
		int counter = 0;
		return source.Select(input =>
		{
		var inputA = new float[input.Length / 2];
		var inputB = new float[input.Length / 2];
		Array.Copy(input, 0, inputA, 0, inputA.Length);
		Array.Copy(input, inputA.Length, inputB, 0, inputB.Length);

			var first = true;
		var prevA = false;
		var prevB = false;
		var direction = 0;
```

```
            for (int i = 0; i < inputA.Length; i++)
            {
                var a = inputA[i] > Threshold ? true : false;
                var b = inputB[i] > Threshold ? true : false;
                var changeA = a != prevA;
                var changeB = b != prevB;
                if (first) first = false;
                else if (changeA || changeB)
                {
                    if (prevA == prevB) direction = changeB ? 1 : -1;
                    else direction = changeA ? 1 : -1;
                    counter += direction;
                }
                prevA = a;
                prevB = b;
            }
            return counter;
            });
        });
        }
}
```

Testing: Reads with extremely high resolution with drifts of roughly 5-10 ticks. Very sensitive to mechanical drift, can add an additional 40-50 ticks of drift. Full tick range: [-300,300].

Bonsai with CF Harp Behavior Board

Use the Harp Breakout board and connect the A and B channels of the encoder to the DI and DIO ports on the breakout board. Connect the breakout board to **Port 2** on the behavior board. Call the quadrature signal reading with the following code in Bonsai:



Device (Harp)

Under "PortName" in properties, select the appropriate COM port that your behavior board is connected to.

BehaviorEvent (Harp.CF)

Set "Mask" to Port 2 and "Type" to Quadrature Counter.

Output the code to a sink node such as WriteToCSV to record the data.

Testing: Reads with high resolution with drifts of roughly 3-5 ticks. Sensitive readings as well and must control for the mechanical drift of the joystick. Full tick range: [-300,300]

PyControl with CF Harp Soundcards

Use the Harp Serial Adapter board and make sure the PyControl board's **Port 1** is connected to the **Input** port on the adapter. Connect the **Output** port on the adapter to the Soundcard's **Serial** port. Plug both USB's into the Soundcard and ensure all appropriate connections with the amplifier and speaker. When complete the sounds can be initiated with the following code.

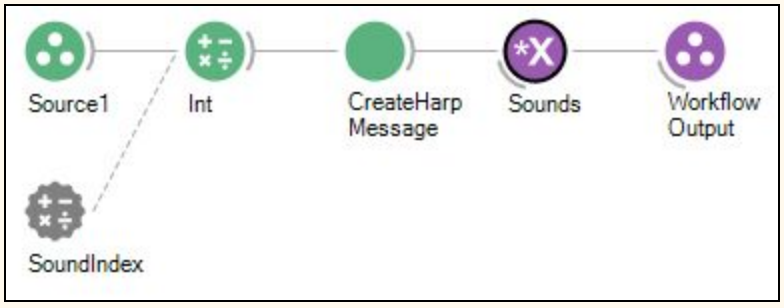
```
#Place the following two lines at the top of your code in definitions

from pyb import UART

import time

#The following three lines must be placed in your code, wherever you wish to initiate a
sound.  Make sure the indentations are properly formatted and the byte number corresponds
with the sound index of the file you wish to play on the Soundcard

uart = UART(4,baudrate=1312500)

uart.write(b'P\x02')                    # send byte 'P' followed by byte 2 for sound index 2

time.sleep(2)                           # wait 2 seconds.
```

Note that sounds can not be added through PyControl, you still need to run the Harp Soundcard software to add sounds and store them in a specific index in the Soundcard. Once you have done that make sure to close the connection with the Soundcard in the Harp software before running the program in PyControl.

Testing: Program would run for a maximum of about 8-10 trials before crashing the PyControl GUI. Multiple noises in succession (1 sound per second) often led to the PyControl GUI crashing or disconnecting as well. If anyone has better success with a different setup, let me know, as my setup all involved USB connections so this may have been due to a bandwidth bottleneck issue.

Bonsai with CF Harp Soundcards

Ensure the Soundcard and Amplifier are connected to the Computer (Requires 2 USB connections from the Soundcard, the connections for the Soundcard to Amplifier, the connections from the amplifier to the speaker and the power supply for both the Soundcard and Amplifier). Once set up, download the PlaySound.bonsai file from the GitHub Repository or create one yourself with the following code.

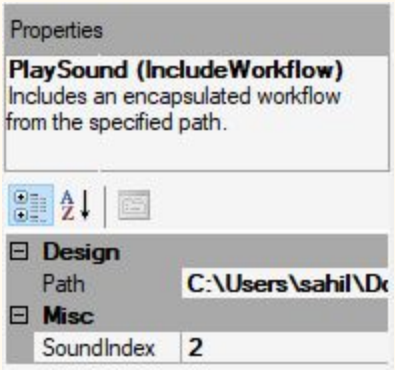


MulticastSubject (Sounds)

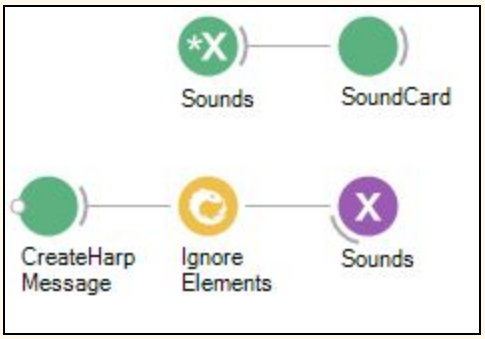
The title of the subject can be anything you assign it as long as it is consistent in the code, similar to a variable.

SoundIndex

Right click on your Int node and you should be able to Externalize Property>Value which will be your Sound Index that you can edit in the top level in the Properties tab as shown to the right.



Now create a GroupWorkflow node that will be titled SoundCard and will interface directly with your Harp Device. Add the following code within the workflow.



SubscribeSubject (Sounds)

Make sure the title is consistent with whatever you named the variable in the previous part (in this case, Sounds).

Device (Harp)

Under "PortName" in properties, select the appropriate COM port that your SoundCard is connected to.

BehaviorSubject (Expressions)

Again, make sure this is named consistently with what you named the SubscribeSubject node and the MulticastSubject node.

Now, by changing the SoundIndex value, you should be able to play any sound that was loaded onto your Soundcard device with the Harp Sound Card software.