Andrew Shanley

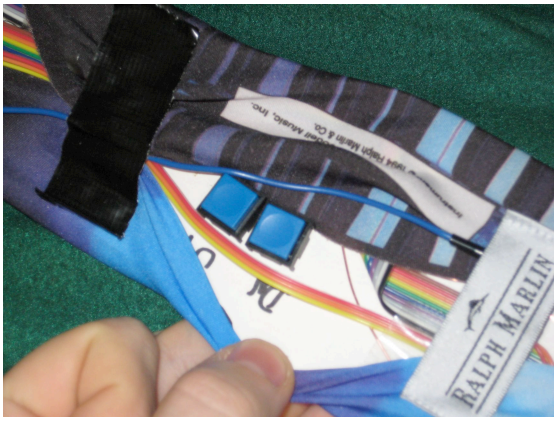
The Piano Key Necktie

The Project:

Inspired by watching *Zoolander* a few too many times, we decided to make a piano key necktie not only highly fashionable, but functional as well. The goal was to make the tie as playable as The tie consists of thirteen keys, giving it a full octave range, two octave change buttons, three modification buttons as well as a slider to control the effect of these mod buttons.

Hardware:

The tie consists of 18 push buttons and a slider mounted on a piece of cardstock that fit inside of the tie. On the front are 16 of the buttons; 13 keys traced to correspond with the image of the keyboard on the tie, and 3 buttons that when depressed will modify different aspects of the sound depending on the combination. On the back of the cardstock are the slider and two buttons, one will toggle the octave up, the other down. To insert the buttons into the cardstock, they were firmly pressed against the cardstock, leaving a slight indent to show where the four prongs of the button lined up. Then using a hammer and a nail the same diameter as the prongs, the holes were punched through and the button inserted. The same was done for the slider. A common ground was soldered to each of

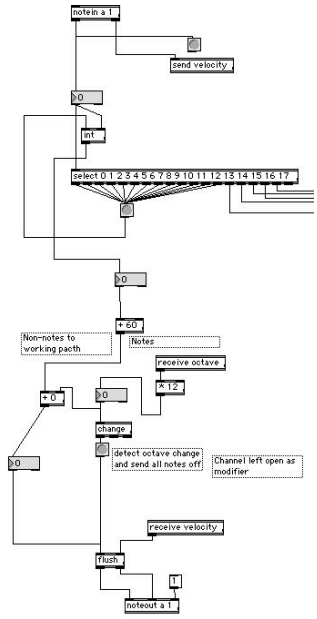


the buttons, and the remaining terminal was soldered to a rainbow colored ribbon cable, making it very easy to keep track of which note corresponded with which color.

Software:

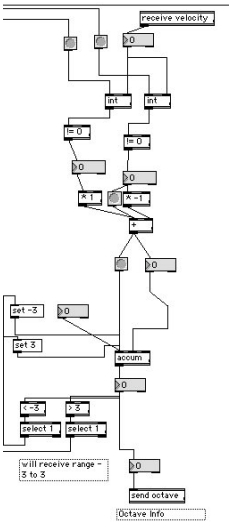
Reading a Note In:

Since there was only one continuous controller being used in this project, we dealt almost exclusively with momentary switches. When a button was depressed, its corresponding note value would be stored in an int, and then sent to a select statement to determine if the button pressed was a note key, an octave key, or a mod key. If it was a note key (values 0 through 12) the select statement fires a bang which then bangs out the value of the note stored in the int. A value of 60 was then added to the note (so zero would be a middle C, and correspond to middle C on the image of the keyboard on the tie). A value of plus or minus 12 multiplied by which octave you are in will be added to the value, bringing you with the final note value to be played.



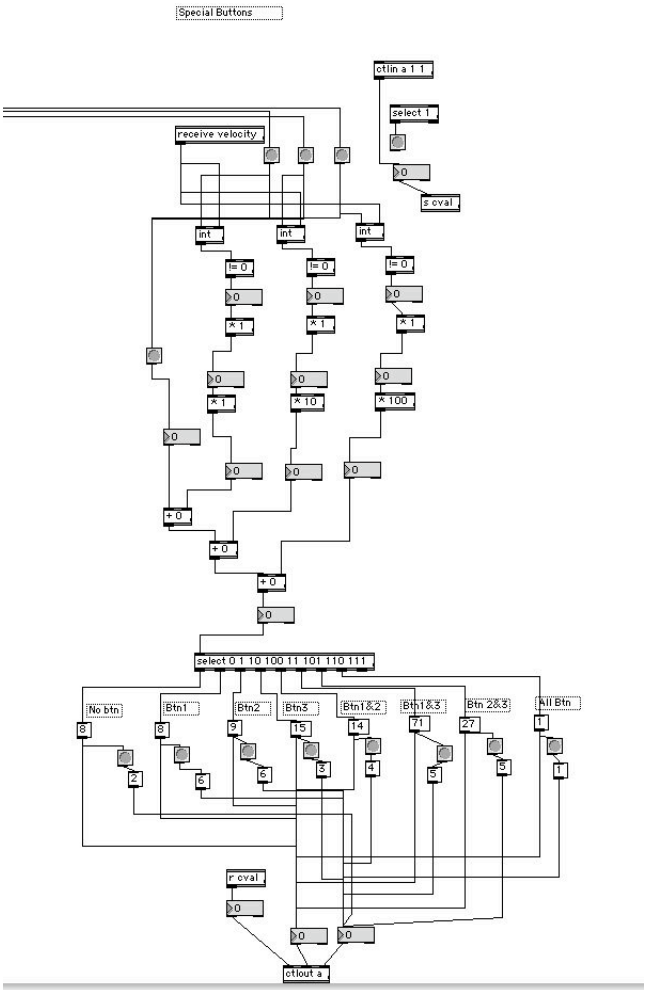
Modifying the Octave:

If the select statement mentioned earlier reads a value of 13 or 14, it then enters a series of logic to ensure there is a octave range from -3 to 3. If a value of 13 is passed (octave up) or 14 (octave down), the select statement fires a bang wired to an int. The int's value is the velocity of the note, and that value is passed to a logic statement when the bang is fired. If the value is not zero (when the note is



pressed, not let go) the resulting value of 1 is then multiplied by either positive or negative 1 (octave up or down). This value is then added into an accumulator, which has a limited range of -3 to 3. This value is then passed out to be multiplied by 12 and added to the original note value.

Mod Buttons:



If the value read by the select statement is a 15, 16, or 17, it then passes along a bang and a value of 1 if the velocity is not zero in the same fashion of the octave switches. One of the buttons was then given a value of 1, 10, and the third a value of 100. This way, when multiple buttons were pressed, there were 8 distinct combinations of button configurations that could be read by max (a value of 0, 1, 10, 100, 11, 101, 110, 111). Each combination affected a different aspect of the sound:

Conclusion:

Despite using a conventional controller design, I feel that it was modified and creative enough to create a really fun and interesting result. If this tie was hooked up to a Miditron (and maybe some pocket speakers) you would essentially have a fully functioning portable keyboard with the ability to change 8 aspects of the sound (only 3 less than the oxygen 8!). Everything ran smoothly, with only a few hiccups along the way. The only "major" problem we encountered was that we could not get the bend sensor to work correctly, and if we had more time that would be the first thing to correct. Also, it would be nice to implement a memory of sorts to store the controller values so you wouldn't have abrupt changes when flipping mod buttons.