Mini-Assignment 1: Arduino

Handed out: Tuesday, 23 September 2014
Due: Friday, 3 October 2014

The purpose of this assignment is to help you become familiar with using Arduino to create digital media interfaces. You will use sensors and actuators to create a simple musical instrument and evaluate its performance.

Required Materials:
- Arduino Experimenter's Kit, including breadboard and speaker
- 8 pushbuttons with 10k resistors
- 1 Force-Sensitive Resistor (FSR) with 10k resistor

Instructions:
You will build a musical instrument that translates the performer's motions, as measured by an FSR and several buttons, to sound.

Step 1: Connect and read the FSR
See the instruction sheet that comes with the FSR for information on how to connect it to the Arduino. You will use the FSR plus one extra resistor to create an analog voltage you can read on one of the analog input pins. Once you've connected the FSR, write a short piece of code in the `loop()` function that reads values from the FSR and prints them to the serial port. Don't forget to call `Serial.begin(9600)` in `setup()`.

Questions (include brief answers in your report): What range of values does the sensor put out? Do the values scale linearly with finger pressure; if not, how would you describe the relationship between pressure and value?

Step 2: Use the FSR value to create a tone
You can remove (or comment out) your `Serial` code at this point. Instead, attach the speaker from your kit and use the value you read from the FSR to control the frequency of a tone created with the `tone()` function. Try scaling and shifting the raw sensor value so that the whole range of pressure covers a useful part of the musical spectrum. For example, what would you need to do cover the two octave range between A3 (220Hz) and A5 (880Hz)?

Questions: What range of frequencies does the speaker reproduce well? Is it easy or difficult to play a melody on the FSR, and why? What might be done to make it easier to play?

Step 3: Use the buttons to make a simple keyboard
This step doesn't use the FSR, but keep it connected: you'll need it again in Step 4.

Attach eight small buttons to eight digital inputs on the Arduino (remember, each one needs a resistor too). Make sure you set `pinMode()` in your `setup()` function. Make each button play a different note (i.e. a different frequency). You can find a list of frequency values in the

`toneKeyboard` sketch in the Arduino examples (File->Examples->Digital; see the `pitches.h` tab). Try these notes to start with: C4, D4, E4, F4, G4, A4, B4, C5. When no buttons are pushed, no sound should play (*hint:* `noTone()`). Don't worry about handling multiple buttons pushed at once.

Questions: Is this instrument easier or harder to play than the one in Step 2? More or less expressive?

Step 4: Add vibrato using the FSR

Vibrato is a technique used by singers and instrumentalists, where the frequency of the note fluctuates periodically up and down by a small amount. In this step, you will use the FSR and buttons together to add vibrato to the keyboard. Use the buttons to select the nominal frequency for the note, as you did in Step 3, but now use the FSR value to vary the frequency up to 5% in either direction. For example, if your note had a frequency of 500Hz to start with, the FSR would give you a range between 475Hz (-5%) and 525Hz (+5%); if your note had a frequency of 1000Hz, then your range would be 950Hz (-5%) to 1050Hz (+5%). You should now be able to add vibrato to each note by periodically varying the pressure on the FSR.

Questions: How do you have to hold the FSR if you want to be able to bend the note both up and down around its centre frequency? How easy is it to play in tune this way? Any ideas for changes that might make it easier to stay in tune?

Special Bonus Step (extra credit): Play a sequenced melody

Change one of the buttons (or add a button) such that pressing it repeatedly plays a pre-composed melody. Each time you press the button, the next note of the tune should play. When the tune reaches the end, the next button press should start it back at the beginning. Play each note for as long as the button is held down; when the button is not pressed, there should be no sound.

Hints: Store the melody in an array, and use an int to keep track of where in the array you are. How do you detect that the button has "just now" been pressed down (as opposed to being held down)?

Your submission should consist of the following:

- Your Arduino code for each step
- Short report (PDF, 1-2 sides of A4) describing your procedure and results, and briefly addressing the questions above
- Quick (1 min.) in-class demo for instructors to verify your instrument works
- *Make a .zip or .tar.gz file of your code and report and submit it via Intranet*

Useful Arduino Functions:

`analogRead()`	Reads a value from an analog input pin
`digitalRead()`	Reads a digital value (HIGH or LOW) from a digital input pin
`tone()`	Generates a tone of a particular frequency on a given pin
`noTone()`	Stops a currently playing tone
`Serial.println()`	Print variables or other information on the serial port

See the Arduino reference (`http://arduino.cc/en/Reference/`) for more details.