ELEC 361 Measurement and Analysis
Lab 7. Fun with Arduino #2 :: Half a Theremin

This **2-week lab** requires you to build a distance-sensing frequency-modulating sound-generating instrument, inspired by the Theremin [5].

First steps with Arduino (common to all Arduino labs)
Background
Arduino is an open-source microcontroller platform that provides flexible, easy-to-use hardware and software. A vast range of 'shields' (boards that can be plugged on top of the Arduino PCB extending its capabilities) are available to implement sensors and interfaces of just about any kind. Programming is via the Arduino IDE (integrated development environment) using the 'Wiring' language that is a cut-down version of C++, to produce a control program called a 'Sketch'.
We will be using the Seeeduino V4.2 that is a breadboard friendly version of the Arduino Uno running the Atmega328 microcontroller.

Pre-Lab Reading
There is quite a bit of reading to do, though most of it is really just skimming through material so you know where to find things:

- For general background information, read the Wikipedia article at http://en.wikipedia.org/wiki/Arduino
- Seeeduino V3.0 blurb and circuit diagram at http://www.seeedstudio.com/wiki/Seeeduino_v4.2#Introduction
- Getting Started with Arduino on Windows at http://arduino.cc/en/Guide/Windows
- Arduino language reference http://arduino.cc/en/Reference/HomePage

Getting started in the lab

- Connect the computer and Seeeduino via USB – this supplies power to the Arduino.
- Launch the Arduino IDE, ensure the correct Arduino board is selected, COM port other than 1, open the blink example, and upload the blink program. (RX/TX LEDs flicker as the upload happens, then you should see the onboard LED blink.)
- Modify the blink program to make the onboard LED blink on for 2 seconds, off for 0.2 seconds.
- Plug the Arduino into the ELVIS breadboard -- careful please – and connect an ELVIS LED to an i/o pin other than 13. Modify the sketch to make the ELVIS LED blink. (Note that a sketch has two parts: The routine “void setup()” is run first and only once. This is where you define variables. The loop “void loop()” routine is run repeatedly after that.)

Half a Theremin Lab

This is a 2 week lab, with 4 parts.

For part 1, you will be using a digital ultrasound transducer to measure distance and displaying that distance with the serial monitor (computer screen).

Part 2 will involve setting up the Arduino as an arbitrary signal generator using the PWM (pulse width modulation) output capability.

In Part 3 you will implement a time-stable numerically controlled oscillator, and verify its operation.

Part 4 requires you to combine these functions, so that frequency of an audio signal generated is proportional to the distance measured by the ultrasound meter, practically making a digital Theremin, see reference [5]. This last step requires some algorithm optimisation to overcome speed limitations of the Arduino, so the instrument sounds groovy.

Overall Objectives

From this lab you will become familiar with:

- Using digital sensors and breakout boards with the Arduino
- Programming an Arduino using the IDE, making use of routines, input/outputs, loops, and serial communication
- The basics of digital to analogue conversion and waveform synthesis
- Timing control using interrupts
- Making kind-of music with half a Theremin

Equipment required

Arduino and programming cable
HC-SR04 ultrasound module
Hookup wires
Audio amplifier and speaker
ELVIS II system including virtual-instrument oscilloscope
Curiosity and a sense of fun

Week 1 Pre-Lab Reading

Tutorial on ultrasound rangefinder and Ping sketch at http://arduino.cc/en/tutorial/ping

Week 1 Pre-Lab Questions

What is the (basic) operating principle of an ultrasound rangefinder?
What is the operating frequency of the HC-SR04 ultrasound module?
How far does (ultra) sound travel in 1ms?
How can you limit the time taken for rangefinding, in the ping sketch, to correspond to a (maximum) distance of 40 cm?

Week 2 Pre-Lab Reading

Wiki page on numerically controlled oscillators (NCO) (aka direct digital synthesis (DDS)) http://en.wikipedia.org/wiki/Numerically_controlled_oscillator. Also see useful tutorials on DDS [1,2].
Tutorial on an Arduino DDS Sinewave Generator at http://interface.khm.de/index.php/lab/interfaces-advanced/arduino-dds-sinewave-generator/. (We will be using a 'corrected' version of this sketch.)

Week 2 Questions

Draw a block diagram of a numerically-controlled oscillator, and describe the main functions.
Suppose your lookup table contains N samples (note: not 2^N) of one cycle of sine, your phase accumulator is n bits wide, and the sample clock has period t. Write a formula that relates value in the frequency-control-word to the frequency of the sine wave produced.
For each of the data types: int, long, unsigned long, double, give the number of bits of storage required for each type, and the range of numbers that can be represented by each type.

Lab Tasks

Take screenshots of your work throughout the lab and hand in your program code with the final report. Preferably, also include a photograph of your setup.

Ultrasound Sensor

1. Connect the ultrasound sensor module to 5V and GND on the ELVIS, and its echo and trig pins to Arduino digital pins 8 and 9 respectively.
2. Upload the (modified) ping sketch and check the measurements on the serial monitor (Tools > Serial Monitor). Display reported delay as well as distance.
3. How accurate is the ultrasound distance measuring device? (A rough measure of error with units is sufficient.)
4. Set the timeout in the pulseIn routine to limit the time delay to 2 ms.

NCO (DDS) with timing by interrupts

5. Why are interrupts useful when making a numerically controlled oscillator?
6. Open the sketch Arduino_DDS_PWM_corrected and read it over so that you understand the purpose of the main functional blocks.
7. Connect a potentiometer (as in the tutorial) so you can vary the voltage at A0, and view the PWM (pulse-width modulation) output at pin 11 using the Scope. (You should simply be able to see something is happening, and that you are changing it; use a 10k, 4n7 low-pass filter if you would like to see a waveform.)
8. View the (approximation to the) sinewave output, and verify frequency.
9. Use your formula for the frequency-control-word directly to set frequency. How finely can you control frequency?

Putting it together

10. Modify the sketch so that frequency is determined by the ultrasound rangefinder. (Start with frequency equals duration in microseconds; you might find it useful to display some diagnostics.)
11. Connect a powered loudspeaker so you can listen to the waveform you generate.
12. Fiddle with the code (e.g., distance-to-frequency function, rate of frequency updating, etc) to make a usable musical instrument. Does pre-computing constants help?

Further Notes

The original Theremin used RF heterodyne coupling (via the hands) to control both pitch and amplitude of the synthesized waveform – see ref [4]. You might consider how you could extend this lab to also control pitch. The Arduino we are using has limited computing power, which limits the quality of the instrument that can be produced. Using an Arduino with greater computing power (such as the Due) could solve this problem, as could using multiple Arduinos of the type we have here (which is cheaper). How could one connect multiple Arduinos to construct a better (perhaps perfect?) Theremin look-alike?

References

[1] - http://www.analog.com/library/analogdialogue/archives/38-08/dds.html
[2] - http://www.analog.com/media/en/training-seminars/tutorials/MT-085.pdf
[3] - NI-ELVIS Series II orientation manual (bound copies are in the lab)
[4] - https://www.youtube.com/watch?v=w5qf9O6c20o
[5] - http://en.wikipedia.org/wiki/Theremin