Waveform generators (also called function generators) are useful for testing and debugging circuits. They can be used to test the frequency response of electronic components like op amps and sensors or to characterize and troubleshoot audio effects boxes and pedals. This waveform generator shield is powered by an Arduino. It outputs four waveforms: sine, triangle, pulse, and saw, each waveform ranges in frequency from 1Hz-50 kHz. The frequency, pulse width, and overall amplitude (gain) of the waveforms is controlled by three potentiometers. Four indicator LEDs let you know which type of wave is currently being sent to the output. This Instructable describes how to put together the Arduino Waveform Generator Shield Kit from Jameco, if you're interested you can also check out my original post about the Arduino Waveform Generator (http://www.instructables.com/id/E7T6UN2H12SVHY8/).

Time Required: 3 hours depending on experience
Experience Level: Beginner

Required tools and parts:
- Soldering Iron
- Solder
- Wire cutter
- Enclosure (wood enclosure in photo was built separately)

Parts List:

<table>
<thead>
<tr>
<th>SKU</th>
<th>Qty</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>119011</td>
<td>4</td>
<td>Momentary push buttons</td>
</tr>
<tr>
<td>286273</td>
<td>3</td>
<td>Linear taper 10k Ω potentiometer</td>
</tr>
<tr>
<td>265077</td>
<td>3</td>
<td>Instrumentation Knob, for 1/4&quot; shafts</td>
</tr>
<tr>
<td>334502</td>
<td>4</td>
<td>White LEDs</td>
</tr>
<tr>
<td>2157183</td>
<td>10</td>
<td>1/4 watt resistor, 220 Ω</td>
</tr>
<tr>
<td>70755</td>
<td>1</td>
<td>Female header socket, 1x8</td>
</tr>
<tr>
<td>103393</td>
<td>4</td>
<td>Male header pins, 1x10</td>
</tr>
<tr>
<td>24133</td>
<td>1</td>
<td>IC, LM386, Op-Amp</td>
</tr>
<tr>
<td>51626</td>
<td>1</td>
<td>IC socket, DIP-8</td>
</tr>
<tr>
<td>2157167</td>
<td>20</td>
<td>1/4 watt resistor, 10k Ω</td>
</tr>
<tr>
<td>691171</td>
<td>10</td>
<td>1/4 watt resistor, 20k Ω</td>
</tr>
<tr>
<td>606820</td>
<td>2</td>
<td>Capacitor, 220μF</td>
</tr>
</tbody>
</table>
Step 1: Header Pins

Solder two groups of 8 and two groups of 6 header pins to the PCB. You will need to clip the 10 position pin headers to the appropriate number of pins. Make sure to solder the header pins to the bottom of the board (the side that does not have any words). Start by soldering down only two points on a row of header pins, one on each side. Then check the alignment of the pins before soldering the rest of the pins down. This way if things are not aligned you can easily reheat one of your soldering points and straighten out part. The shield should fit nicely on an Arduino and will also help keep the pins aligned.
Step 2: Tact Switches

Press fit four tact buttons onto the side of the PCB; solder all four leads to the pads of the PCB.

Step 3: 8 Pin Socket

Solder one 8 pin socket to the part of the PCB labeled U2. Follow the tips in Step 1 for soldering the socket on straight.
Step 4: 220µF Capacitors

Solder two 220µF electrolytic capacitors inside the large circles printed on the PCB. Make sure to line up the negative lead of the capacitor with the "-" label on the PCB.

Step 5: 10k Ω Resistors

Solder (11) 10k Ω resistors to the PCB. Four of the resistors go in the spots labeled R1-R4 and seven go in R9-R15. Remember, the orientation of the resistors does not matter.
Step 6: 20k Ω Resistors

Solder nine 20k Resistors in the spots labeled R16-23, R22, and R24.

Step 7: Voltage Divider

Solder one 20k Ω resistor in the spot labeled R28 and one 2.2k Ω resistor in the spot labeled R29. (Note: the color bands of the 2.2k resistor in the picture are wrong; they should be red / red / red / gold).
Step 8: 220 \( \Omega \) Resistors

Solder four 220 \( \Omega \) resistors in the remaining spots labeled R25-26, and R7-8.

Step 9: Potentiometers

Solder the three 10k \( \Omega \) potentiometers to the spots labeled AMP, PULSEWIDTH, and FREQUENCY.
Step 10: LEDs

Solder four LEDs to the PCB as shown in the image above. Make sure to line up the flat edge of the LED with the mark on the PCB.

Step 11: 9V Battery Snap

Solder the battery clip to the Vin and GND of the PCB as shown in the image above. Also solder leads or female header sockets to the ground and signal pins of the board, these are your outputs.
Step 12: LM386

Press the LM386 into the 8 pin socket. Make sure that the top of the IC (pins 1 and 8) are facing toward the potentiometers.

Step 13: Firmware

Download Arduino IDE and upload the code at the bottom of this step onto the Arduino. The code uses a timer interrupt at a frequency of 100 kHz to send new data out to the digital to analog converter (DAC). The rest of the code monitors the state of the buttons and knobs and adjusts variables accordingly. Since the interrupts occur at such a high frequency, I had to keep the interrupt routine, the piece of code encapsulated in the ISR(TIMER1_COMPA_vect){} as short as possible. Time intensive operations like mathematical operations with floats and using the sin() function take too much time to complete. I used several work arounds to get by this.

For triangle and saw I created the variables sawByte, triByte, sawInc, and trilnc. Every time the frequency changed I calculated the amount that the triangle and saw function would have to increment at a sampling rate of 100 kHz:
triInc = 511/period;
if (triInc==0){
    triInc = 1;
}
sawInc = 255/period;
if (sawInc==0){
    sawInc = 1;
}

then all the needed to be done in the interrupt routine was some simple math:

case 1://triangle
    if((period-t) > t);
        if (t == 0){
            triByte = 0;
        }
        else{
            triByte += triInc;
        }
    }
else{
    triByte -= triInc;
}
if (triByte>255){
    triByte = 255;
}
else if (triByte<0){
    triByte = 0;
}
wave = triByte;
break;

case 2://saw
    if (t=0){
        sawByte=0;
    }
else{
    sawByte+=sawInc;
}
wave = sawByte;
break;

For the sine function, I wrote a simple python script which outputs 20000 values of 127+127\sin(x) for one complete cycle:

```python
import math
for x in range(0, 20000):
    print str(int(127+127*math.sin(2*math.pi*x*0.00005)),)+str(","),
```

I stored this array in the Arduino's memory called sine20000[] and recalled the values I needed to send to the DAC. This is much faster than calculating the values individually.
Step 14: Output

Turn up the gain knob and attach a nine volt battery to the battery clip (the LM386 needs 9V to work properly). Hook up the function generator to an oscilloscope. Test out each of the waveforms and adjust the frequency and gain to make sure they are working properly. Switch the output to pulse and check if the pulse width modulation knob works (figs 4-6). The LEDs corresponding to each waveform should light up as each waveform is selected.

You will notice that the pulse wave is the only wave which truly ranges from 1Hz to 50 kHz. Since the sampling rate is 100 kHz, the sine, triangle, and saw waves start to become somewhat unrecognizable at about 25 kHz (they are only comprised of 4 samples per cycle- 100 kHz/25 kHz). The saw and triangle waves only go down to about 100Hz, this is because the values of trilnc and sawlnc get so low that they are rounded to zero below this frequency. The sine wave reaches all the way to 1 HZ but the resolution stays the same for anything under 5Hz, since the Arduino only has enough memory to store about 20 thousand samples.
Step 15: Troubleshooting

In case your project is not working try the following:

**Check for continuity:** Review the schematic and board layout above and check your connections for continuity. It's possible that you could have accidentally short circuited something or you may have a loose connection. Especially check that everything which should be grounded is connected to the Arduino's ground. Make sure that solder is flowing on both the top and bottom of the board on the ground pin of the output as there are connections on both sides of the board.

**Check the path of the wave:** Try to pinpoint the problem area by checking various junctions of the PCB with an oscilloscope. If you probe either end of R27 you should see the waveform oscillating between 0 and ~5V. At the junction between R28 and the first 220µF capacitor, you should see the same wave centered around 0V. Next check pin 5 of the LM386, again, you will see the waveform centered around 0V.

**Reheat your solder joints:** If you just can't find anything wrong right by inspection, it's a good idea to reheat all your solder joints one by one so that you are sure they are nicely connected. This is a great strategy for troubleshooting any PCB that you might work on in the future.