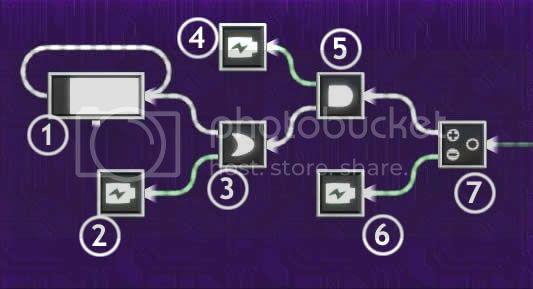
**4. A "Simple" Example**

So far, all we've done is look at the basic tools and their functions and it doesn't seem like much could be done with them - really we've only got a few sensors to detect the state of the world, and a few basic operations we can carry out and that's all there is to our analogue toolset... But when you think about it, this is actually more functionality than the individual components we used to use for logic in the old game and look what we achieved with that.

Next instalment I'll actually put together a couple of neat microchips that will allow us to expand this toolset with more functions, but as this is a bit long and technical so far, I don't want to do any in-depth examples, so I'll finish off with a simple waveform manipulation circuit I dreamt up, which is shown below (click for higher res):



***fig 4.0.1.*** *Wavform Filtering Circuit*

**4.1. Signal sources**

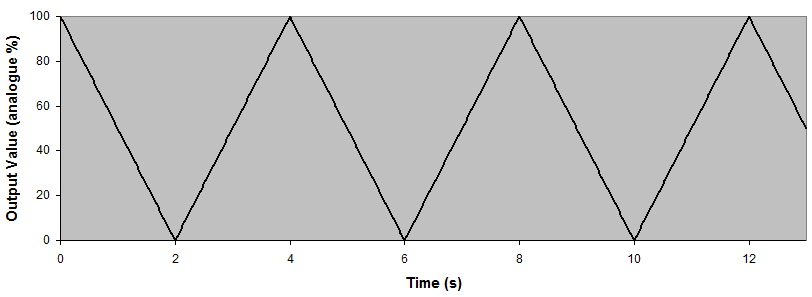
In this case, we're using the timer (labelled 1 in fig 4.0.1) and some batteries (2, 4, 6) as the signal sources. The timer is configured with the following settings:

Target Time: *2s (note, this is arbitrary)*

Input Action: *Forwards / Backwards*

Invert Output: *Yes*

This set-up will produce a triangle wave, as looping the inverted output back to the forwards/backwards input causes the timer to fill up linearly, then empty linearly, then repeat. I don't want to go too much into exactly why this is, but that is what it does, as per the graph below:



***fig 4.1.1.*** *Triangle Wave (System Input)*

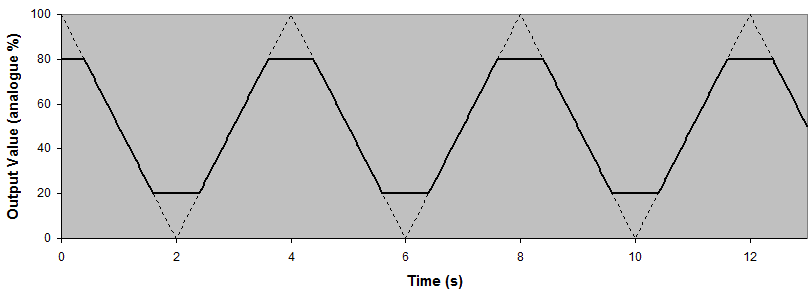
Note that the period of the signal is twice the Target Time setting (as the timer must fill up for 2 seconds and then empty for 2 seconds).

**4.2. Signal Limiting**

As discussed above, some of the simplest operations we can do on analogue signals is min() and max(), using ANDs and ORs. We can use this to achieve some boundary limiting on the signal, preventing the signal going above 80% or below 20%.

So taking the output of the timer and using an OR gate (labelled 3) with a battery (2) set to output constant 20% will give us a max function, which leads to a *lower* boundary. To clarify: if the timer output is greater than 20, the output of the OR gate is equal to the timer, but if the timer drops below 20, then the OR gate will output 20.

After this we do the same thing with an AND gate (5) to do a min() with an 80% battery (4), to give an upper limit, and the resulting output of the AND gate is shown below (with the original wave in dotted lines). You can see that the timer's output is ignored when it exceeds the minimum and maximum bounds that we have set for it:

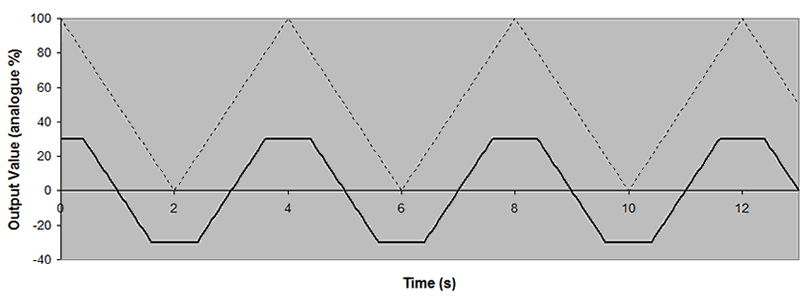


***fig 4.2.1.*** *Trapezoid Waveform (triangle with boundary limiting)*

Conceptually it is a little confusing that we use a min() function to create a maximum value, but that is the way of the world, I'm afraid - it's certainly not my fault!

**4.3. Removing the DC Offset**

Sometimes, especially with waveforms, it's useful to have them centred around 0, and to do this we need to shift the whole wave down by 50 (the point at which it is currently centred - its average value). This is a simple subtraction operation using a combiner (7) and a battery set to 50% (6) and results are as shown:



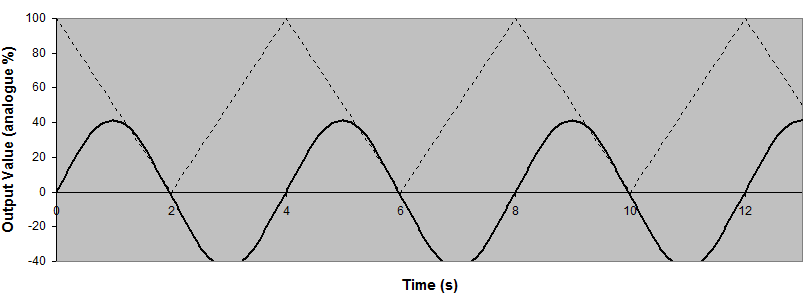
***fig 4.3.1.*** *Trapezoid Waveform with DC component removed*

**4.6. Extending the concept**

The advantage of centring the signal around zero is that over time, the value averages to zero - if you were to put that signal into a mover, it would move forwards a bit, then backwards a bit, then repeat, but it wouldn't drift from its original location. The same is true if you put the signal into a timer with speed input.

The timer with speed input actually works out as being an integration of the input signal against time. I'm not going to explain why - those of you with enough calculus to understand why can probably work it out on your own. One reason integration is good is that it can quite often give us curvy waveforms, and curvy is nice, for all manner of reasons Wink

Integrating our trapezoidal waveform (fig 4.3.1) actually turns out to be a rough approximation of a sine wave, as shown below:



***fig 4.3.1.*** *Trapezoid Waveform Integrated to Approximate Sine*

**Note:** This is only an approximation of a sine wave, I *think* it's possible to produce completely accurate sine and cosine signals, in the range -100% to 100%, but I haven't tried out that mechanism yet as it's a weird complex loopback system, that makes my head hurt to think about it.

To achieve this result I have actually fed the signal into timer starting at 50% and then deducted 50% from the output value, to recentre the signal around zero. This is shown below as an extension of the original image (components 8 and 9 provide this extra processing - the 50% battery at 6 is reused as shown):

