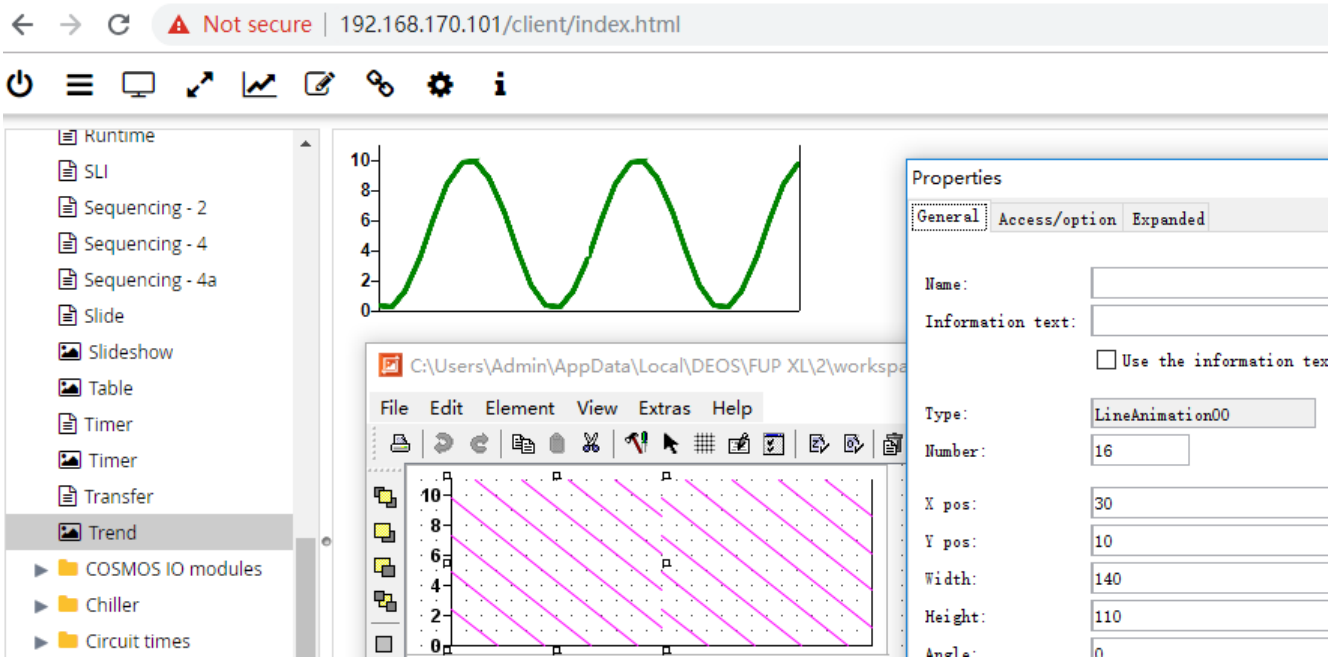
TT211106 – FUP - Dynamic Chart Range

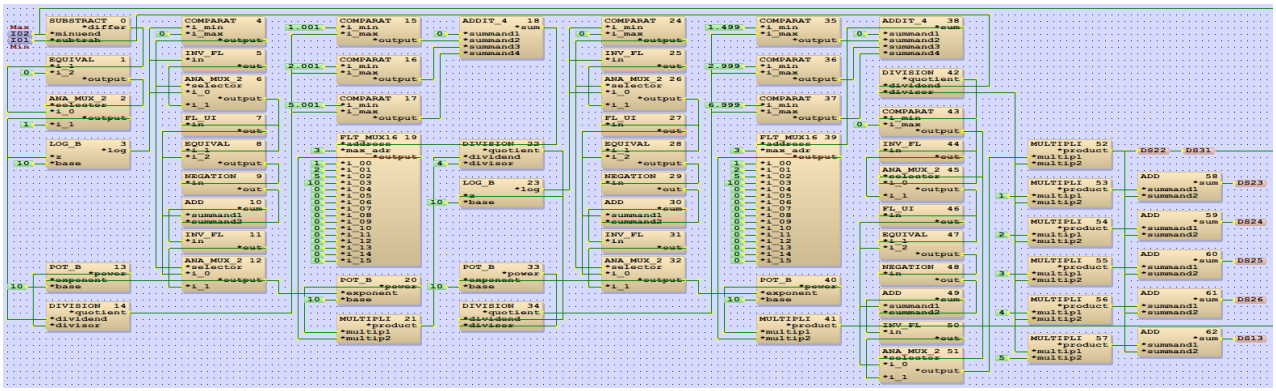
1. In TT200401, we show you how to create a simple line chart (trend graph) in FUP using the “LineAnimation00” graphic element.



2. In that example, we use a fix range of 0-100 in the chart. Sometimes people like to use dynamic range that change based on the min. and max. values. There are many algorithms from the Internet, and in this document, we will base on this algorithms (Nice Label Algorithm for Charts).
3. Below is the principle of the logic.

> When creating a graph by computer, it is desirable to label the x and y axes with "nice" numbers: simple decimal numbers. For example, if the data range is 105 to 543, we'd probably want to plot the range from 100 to 600 and put tick marks every 100 units. Or if the data range is 2.04 to 2.16, we'd probably plot a range from 2.00 to 2.20 with a tick spacing of 0.05. Humans are good at choosing such "nice" numbers, but simplistic algorithms are not. The naïve label-selection algorithm takes the data range and divides it into n equal intervals, but this usually results in ugly tick labels. We here describe a simple method for generating nice graph labels.
>
> The primary observation is that the "nicest" numbers in decimal are 1, 2, and 5, and all power-of-ten multiples of these numbers. We will use only such numbers for the tick spacing, and place tick marks at multiples of the tick spacing...

4. You can find the source code on the website. Below is the implementation in FUP.

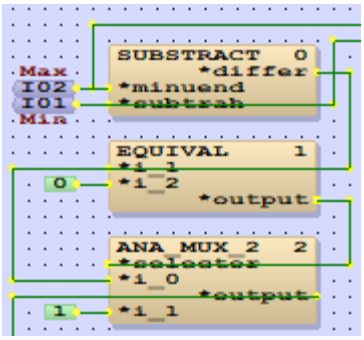


5. This looks complicate, right? Let's take a look one by one and by the end of the document, you should be able to learn how to build complex logic and calculation in FUP.

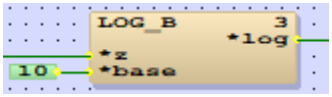
   First, we compare the min and max values. If they're the same, then we set the range to be 1. Otherwise the range, is (max – min).

   After that we do this to normalize the range withing 10.

```
exponent = Math.floor(Math.log10(range));
fraction = range / Math.pow(10, exponent);
```



6. This is the "log10" function in FUP.



7. This is the "floor" function.

Round down value

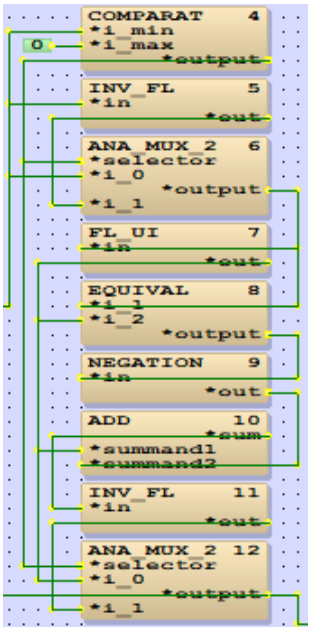
Rounds *x* downward, returning the largest integral value that is not greater than *x*.

   This is a bit complicate as we don't have this function in FUP. So we use the "FL_UI" module. This works well for positive number.

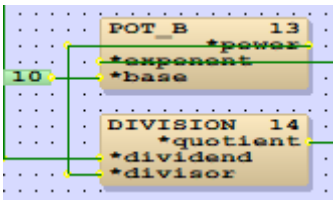
   For negative number, we use "INV_FL" module to convert it to positive number. The "COMPARAT" and "ANA_MUX_2" modules are used to send always positive number to the "FL_UI" module.

   If the number is negative, we need to add 1 to it and then convert it back to negative number. The "EQUIVAL", "NEGATION", "ADD" and "INV_FL" modules did the tricks.

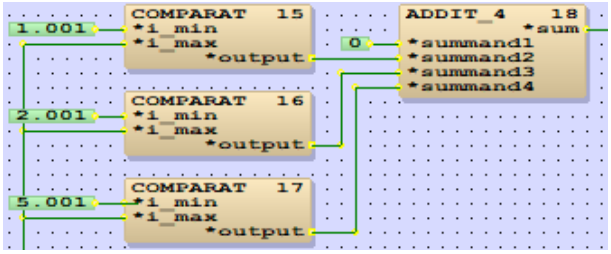
   The last "ANA_MUX_2" module select the correct value (either positive or negative) to the output.



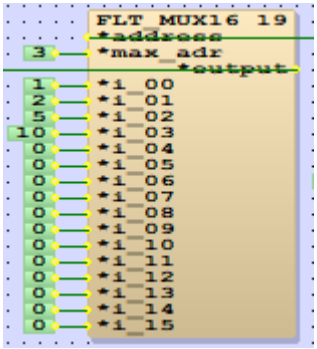
8. This is the "Power" function, and the range is divided by this number to create the normalized value for the range.



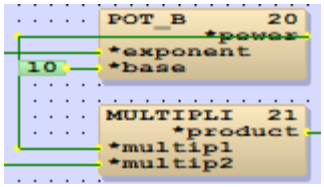
9. Now we compare the normalized range and see if they're less than or equal to either 1, 2 or 5. This will then output as 0, 1, 2, or 3 respectively.



10. Based on the above output, the "FLT_MUX16" module will send the value 1, 2, 5 or 10 to the output.

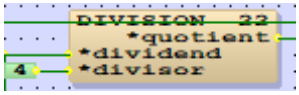


11. Finally, the “nice” range of the chart is returned by the following calculation.



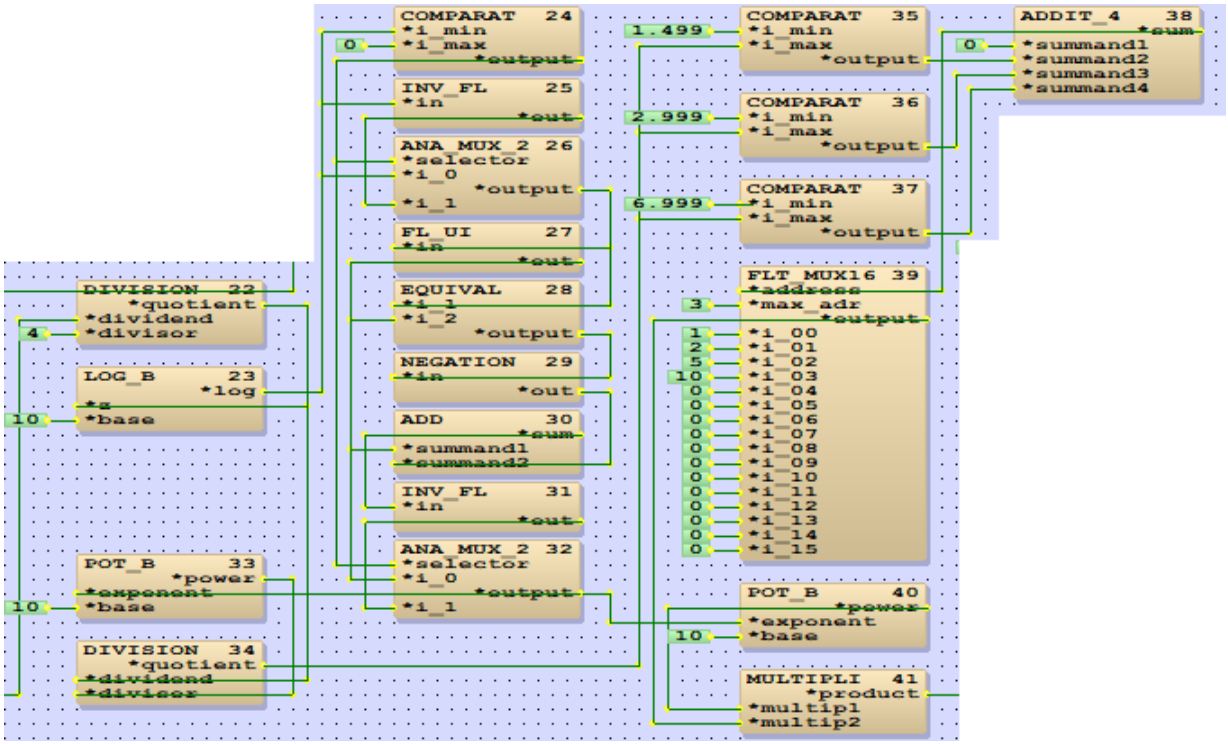

```
return niceFraction * Math.pow(10, exponent);
```

12. Now the “nice” range is found. Then we’re going to find the “nice” tick spacing. In our example, we use 5 ticks, so we divide the range by 4.




```
this.tickSpacing = niceNum(range / (maxTicks - 1), true);
```

13. Similar logic and calculation are used to calculate the “nice” tick spacing. The only different is we check the normalized tick spacing if they’re less than 1.5, 3 or 7, and then select the “nice” tick spacing of 1, 2, 5 or 10.



14. Then the “nice” minimum value is calculated by the below formula. Now you should be able to find out how to do it in FUP.

```
this.niceMin =
  Math.floor(minPoint / tickSpacing) * tickSpacing;
```

15. There are 2 ways to calculate the “nice” maximum value. This first one is calculated by the below formula. This one always calculates the best maximum value (closest to the raw max. value), but the number of ticks may change (i.e. not always 5 ticks).

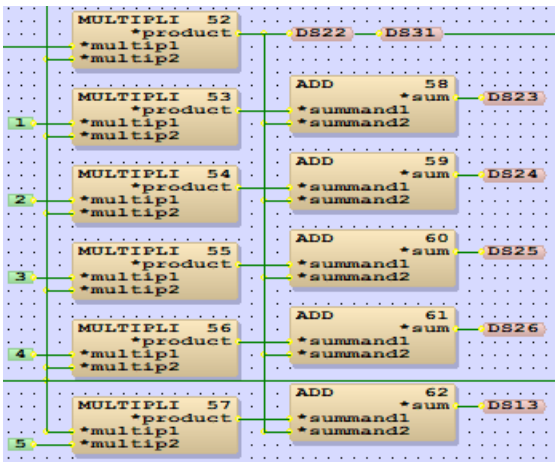
```
this.niceMax =
  Math.ceil(maxPoint / tickSpacing) * tickSpacing;
```

16. Another way is like this. This one will always give you 5 ticks based on the “nice” range and minimum value. But the maximum value will sometimes become larger than we expect because the tick number is fixed.

```
this.niceMax = this.niceMin + tickSpacing * (maxticks - 1); // Always display maxticks
```

17. Since in FUP, it’s not easy to build graphic with changing tick number, so we need to fix it first. Because of this, both methods have their own pros and cons.

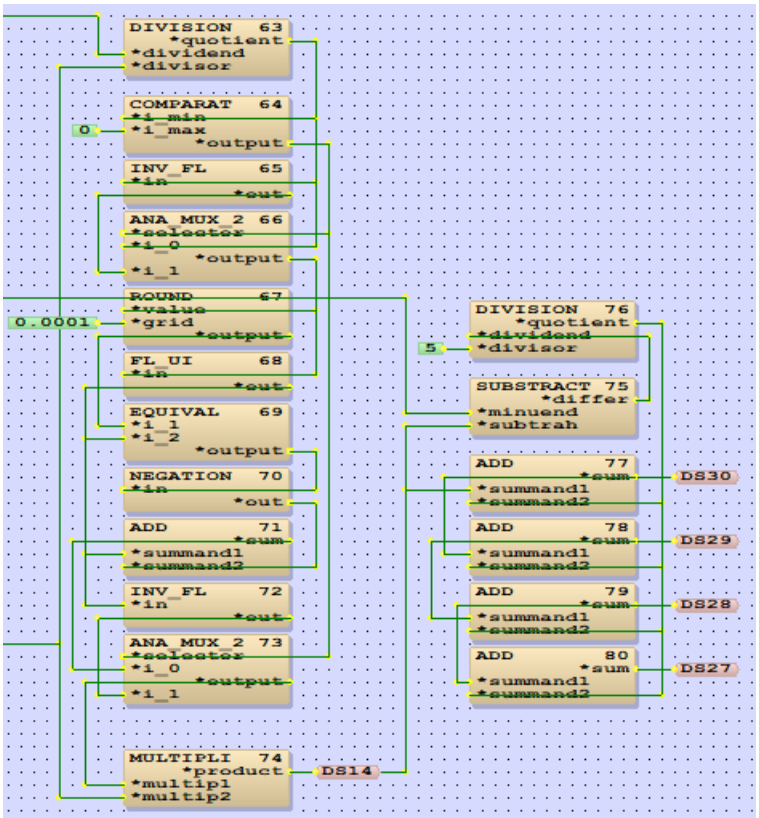
18. The later method is easier as we only need to add the “tick” spacing to the “min” value, and then add it 5 times to come up with the “max” value.



19. The former method is a bit more complicate as we need to use the “ceil” function, which is also not available in FUP.

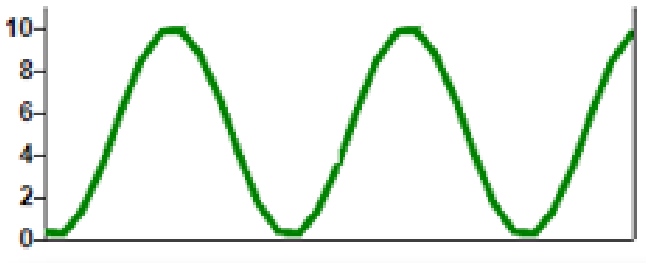
The ceil function returns the smallest possible integer value which is equal to the value or greater than that.

20. The “ceil” function is similar to “floor” function, so here are FUP modules for the calculations.



21. Now, let’s take a look at the outputs. The values can then be used in your chart.

Min Value	0.00	0.00	0.00
Max Value	92.60	20.00	20.00
		40.00	40.00
		60.00	60.00
		80.00	80.00
		100.00	100.00



22. Now check the output for the 2 methods with different min. and max. value. You can see the former method (values on the right) always give you a better max. range, but the tick spacing is not always “perfect”. I’d suggest to use the former method as it’s more “correct”.

Min Value	0.00	0.00	0.00
Max Value	0.83	0.20	0.20
		0.40	0.40
		0.60	0.60
		0.80	0.80
		1.00	1.00

Min Value	156.00	150.00	150.00
Max Value	268.00	200.00	180.00
		250.00	210.00
		300.00	240.00
		350.00	270.00
		400.00	300.00

Min Value	2126.09	2120.00	2120.00
Max Value	2189.26	2140.00	2136.00
		2160.00	2152.00
		2180.00	2168.00
		2200.00	2184.00
		2220.00	2200.00

Min Value	0.00	0.00	0.00
Max Value	2189.26	1000.00	600.00
		2000.00	1200.00
		3000.00	1800.00
		4000.00	2400.00
		5000.00	3000.00

Min Value	-56.30	-100.00	-100.00
Max Value	126.40	-50.00	-50.00
		0.00	0.00
		50.00	50.00
		100.00	100.00
		150.00	150.00

23. So, now you can build a line chart with automatic range based on the min. and max. values of the sensor values dynamically.