Delta-T Devices

Step-by-step Tutorial

Implementing & testing Dew-point and Wind chill algorithms with DeltaLINK 3 Script Editor

Introduction:

The purpose of this paper is to demonstrate application of Delta-t's Script Editor to potential GP2 users. This short tutorial describes the implementation of some common metrology functions; namely dew-point and wind chill factor

Little or no 'programming' experience is assumed, but basic math skills and operation of a 'scientific' pocket calculator has been assumed.

It is not this papers intention to discuss the suitability of a given formula for a particular application, but rather convey an approach to implementing, testing (via simulation) and deploying custom functionality to a GP2 Logger.

Dew-Point calculation:

Consider a GP2 based weather station, where a dew-point calculation maybe desirable. A commonly used approximation has been selected for this purpose; (see references).

The Magnus formula [Sonntag] relates saturation vapour pressure and dew point at temperature T (°C).

Saturation vapour pressure EW (hPA) over water:

$$EW = \alpha \cdot \epsilon \left\{\frac{\beta.T}{\lambda.T}\right\}$$

For temperatures between -45 and +60°C 'Magnus' parameters α=6.112 hPa, β=17.62 and λ=243.12°C can be applied. From the equation above, dew-point temperature (°C) can be derived from vapour pressure:

$$Dp = \frac{\lambda * ln\left\{\frac{E}{\alpha}\right\}}{\beta - ln\left\{\frac{E}{\alpha}\right\}}$$

Applying relative humidity (RH%) to above equation, i.e. E = RH% * EW / 100, dew-point (Dp) can be calculated from temperature and humidity as follows:

$$Dp(T, RH) = \frac{\lambda \cdot \left\{ln\left\{\frac{RH}{100}\right\} + \frac{\beta \cdot T}{\lambda + T}\right\}}{\beta \cdot \left\{ln\left\{\frac{RH}{100}\right\} + \frac{\beta \cdot T}{\lambda + T}\right\}}$$

This can be simplified into a pair of equations suitable for implementation by the script editor:

(Note that H is an intermediate result required by second equation to calculate dew-point)

H = (Ln(RH/100) + (17.62*T) / (243.12+T) **(1)**

Dp = 243.12 * H / (17.62-H) **(2)**

Where:

RH = Relative humidity (%)

T = Temperature (°C)

Dp = Dew-point in (°C)

For this example, the combined temperature and humidity sensor (RHT2nl) is used, having been previously characterised for the GP2 sensor library and tested in the field.

As a general rule it is always worth writing the problem down first, clearly defining the required outputs, e.g. Control relays, calculations performed and results to be logged.

Without the distraction of using a computer, methods for simplifying the problem and testing various calculations with the simulator often become apparent. In this case the process went as follows:

Q. How often should sensor readings be taken?

(1 minute for testing, but as temperature and humidity change relatively slowly, 10 minutes is more than adequate for the final application)

Q. Is the 1 minute 'logging' an issue for RHT2nl sensor; e.g. self-heating?

(No, but data memory will fill up faster in final application)

Q. Should 'raw' sensor readings (e.g. mV and ohms) be logged?

(Yes, for both testing calculations and use by other routines, e.g. wind chill calculation)

Q. Can and how should 'diagnostics' be done to verify script functioning correctly?

(Yes, values can be assigned to temperature and humidity parameters, rather than 'real' sensor values from simulator or actual hardware)

Q. Can equations **(1) & (2)** be written in form for easier script implementation?

(**Yes** – potentially make scripts easier to understand by others at a later date)

Adapting dew-point equations:

Consider equation (1):

H = (Ln(RH/100.0) + (17.62*T) / (243.12+T))

Breaking it down into small steps:

eqa = Ln(RH/100.0)

eqb = 17.62 * AirTemp

eqc = 243.12 + AirTemp

H = eqa + eqb / eqc

Similarly for equation (2):

Dp = 243.12 * H / (17.62-H)

Breaking it down into small steps:

eqd = 243.12 * H

eqe = 17.62 – H

Dp = eqd / eqe

Log the calculated dew point:

Record(Dp)

Fig 1.0 shows above equations or 'code' as entered into DeltaLINK script editor

What is the above code doing?

eqa, eqb, eqc, eqd, eqe, Dp and H, are known as 'variables'. Their names are assigned to values (numbers) and in some instance other variables.

For example;

eqa = 10.0 - assigns the value of 10 to variable named eqa

eqb = eqa - assigns the value of eqa to eqb, in this case 10

Note that the value 10.0 has been allocated to eqa rather than just 10. This indicates the number (and variable), are decimal (or 'floating point'), i.e. 10.13, 10.23 etc. rather than Integer or 'whole' numbers, e.g. 10, 20, 30. By default numbers are decimal. Consult the script editor help system for further details.

RH and AirTemp are 'reserved variables', i.e. they are used by the Script Editor to contain readings from actual sensors, the RHT2 in this case. Replacing either or both of these variables with fixed values provides a convenient method for checking the algorithm with the simulator.

Variable Dp (dew point) is made equal to the result of eqd divided by eqe.

The statement 'Record(Dp)' actually records the value of Dp to memory for subsequent transfer to a computer for additional processing such as graphing or statistical analysis.

How often these equations are evaluated is determined by the application. For this example a new dew-point value is required to be calculated every minute.

Interval or 'repeat rate' is set from within the 'Scripts' list of DeltaLINK program editor, see Fig 2.0. Similarly, 'raw' sensor data can be optionally logged at a rate set from within 'Recordings' list.

'Lists' provide quick access to various sections of the editor. Particular attention is drawn to 'Measurement' where sensors are selected and configured. The 'Outputs and variables' list summarises all user created variables, plus those used to indicate state of control relay; i.e open or closed.

Simulating code:

Having created the script it can be transferred to the simulator application and executed in the same manor DeltaLink does with actual hardware. Simulator sensor readings are 'pseudo random'. They are intended to simulate a temperate maritime climate at 51 deg latitude, (Northern hemisphere); and typical for the UK.

Fig 3.0 is a screenshot from simulation of the above script code. The plotted dew-point temperature (Dp) having being calculated from simulator generated RH and AirTemp values (also plotted).

Verifying whether the script generates sensible results can be achieved in a number of ways. Possibly the easiest being to read simulated RH and temp values from the simulator output graph, and applying them to the original equations (1) and (2). This 'manually' calculated value should be identical to the script calculated dew-point (Dp) shown on the graph.

A flexible and potentially more accurate approach is illustrated by screenshot shown in fig 4.0. Simulated RH and AirTemp are still plotted but Dp is now a horizontal line at approximately -8.8°C. This is due to variables RH and AirTemp (underlined in code above) being replaced with 'constant' values; e.g. RH=10 and AirTemp=25. Future script iterations might also plot these 'constant' values, to remind users that in this instance Dp should also be constant.

This process of iteratively writing and testing scripts allows quick and extensive testing of potentially complex measurement and control algorithms, before deployment on actual hardware in the field.

Fig 1.0 Screenshot from Script Editor showing 'dew-point' algorithm:

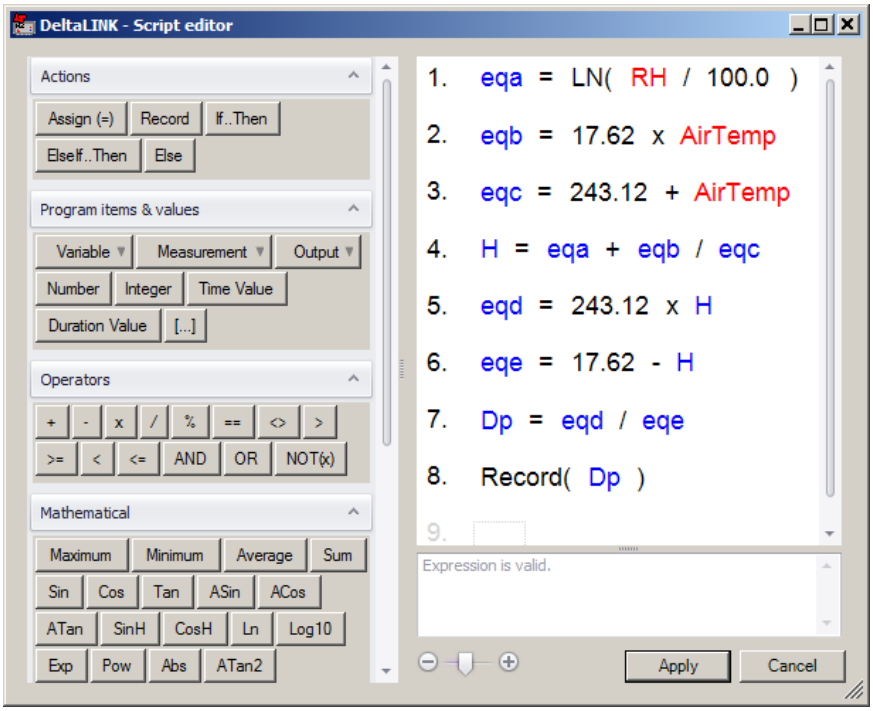


Fig 2.0 Screenshot from Script Editor (top level):

- Setting script 'execution' rate parameter from within 'Scripts' list; (Repeat rate)
- Sensors (RHT2 in this case) are selected from within 'Measurement' list
- Logging rate for raw sensor data set from within 'Recordings' list; (Recording rate)

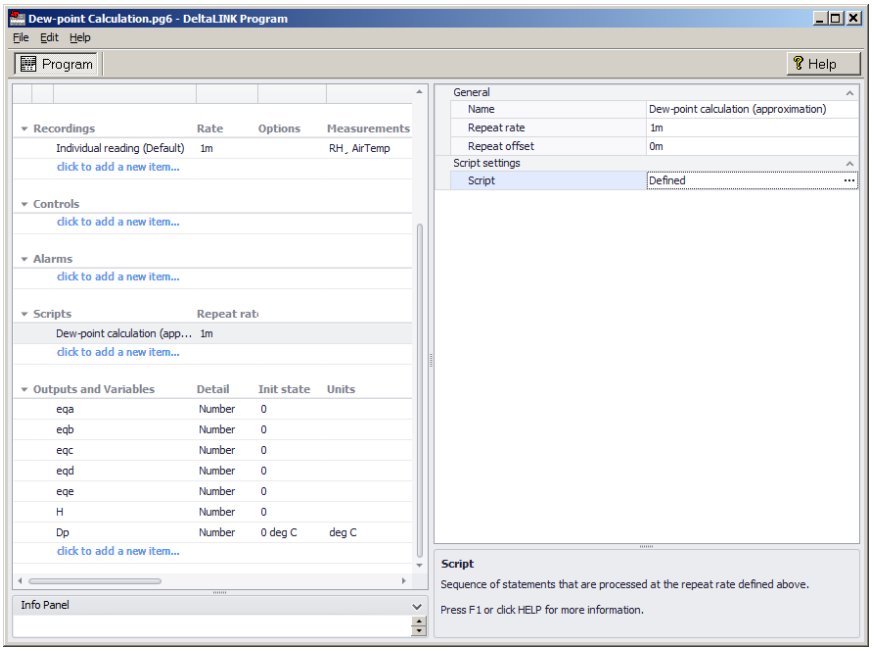


Fig 3.0 Screen shot from simulation of 'dew-point' algorithm:

- RH and AirTemp values are generated by the simulator
- Dew-point temperature (Dp) calculated by equations created in script editor

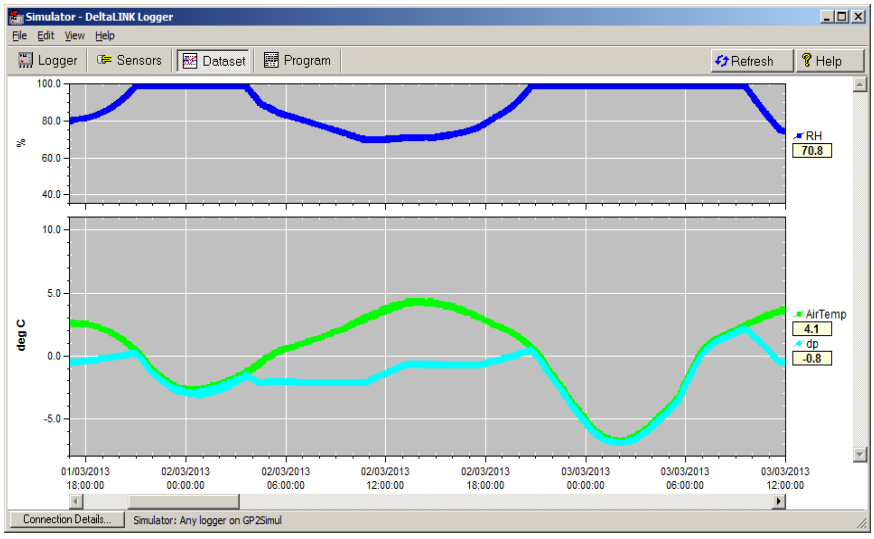
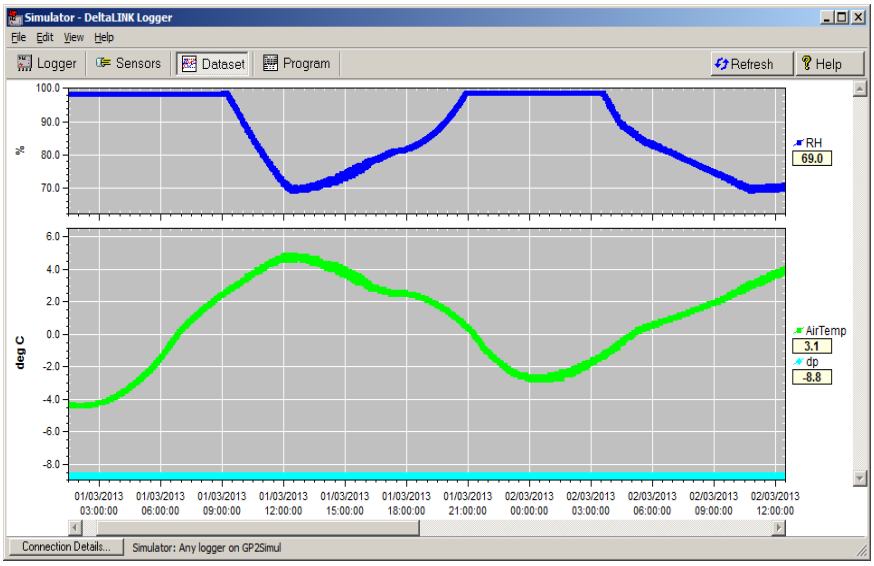


Fig 4.0 Screen shot from simulation of 'dew-point' calculation:

- RH set to 10% and AirTemp 25°C within script to calculate a constant Dew-Point (Dp) of -8.8°C
- Simulated RH and AirTemp values shown on graph are not used



References:

Sonntag D; Important New Values of the Physical Constants of 1986, Vapour pressure Formulations based on the IST-90 and Psychrometer Formulae; Z. Meteorol., 70 (5), pp.340-344,1990.

Hardy B; Thunder Scientific Corporation, Albuquerque, NM, USA. The proceedings of the Third international Symposium on Humidity & moisture, Teddington, London, England, April 1998

Sensiron; 'SHTXX' application note: Dew-point Calculation.

Wind chill calculation:

Introduction:

'Wind chill', or 'wind chill' factor is the perceived drop in temperature felt by exposed parts of a body, resulting from a cold air mass flowing across it; i.e. due to wind. The faster the wind speed, the more rapidly exposed surfaces cool.

In the case of humans (and most biological organisms), the natural physiological response is to maintain surface (skin) temperature within an acceptable range to avoid adverse effects.

Attempting to maintain a given surface temperature in an area of fast heat loss can both give an impression of lower temperatures and an actual greater heat loss. This can increase the risk of effects such as hyperthermia and frostbite.

For inanimate objects such as buildings and machinery, the effect of wind-chill results in warm surfaces being cooled to an ambient temperature more quickly. But whatever the wind speed, surface temperature cannot be lower than ambient.

However, a wet surface e.g. water on a structure, or person wearing wet cloths can drop below ambient due to loss of latent heat from within the water.

Quantifying wind chill in 'real time' thus is a particularly useful function for weather stations and a pretty good indication of weather severity.

Model:

Unfortunately there is no universally accepted formula for calculating wind chill. One of the early methods, (Siple-Passel) was based on observing how quickly bottles of water froze in Artic winds, where more modern techniques have focused on effects to human physiology.

In 2008 the UK adopted the JAG/TI (Joint Action group for Temperature Indices) system, a method used by Canada, the US and the Netherlands.

This model specifically;

- Calculates wind speed at a height of @1.52 metres (typical height of adult human face), based on corrected readings from 'standard' anemometer height of (@10 metres)
- Applies a human face model
- Uses modern heat transfer theory; (heat loss from body to its surroundings, during cold/windy days)
- Reduces calm (no wind') threshold 4.8 km/h
- Assumes worst case scenario for solar radiation, (i.e. clear night sky)

Results from this model can be approximated within one degree from the following formula:

$$Twc = 13.12 + 0.6215Ta - 11.37V^{0.16} + 0.3965TaV^{0.16} \quad (3)$$

Where: Twc is the wind chill index (in Celsius), Ta is the air temperature (in Celsius) and V is the wind speed (km/h) measured at 10 metres (anemometer height)

Practical implementation of the formula:

As with dew-point example consider the target application before starting to write any control scripts:

Q. What sensor types are required?

(AN1 for wind speed and 10K thermistor for temperature measurement)

Q. What are the default measurement units for selected sensors?

(meters/second for anemometer and degrees Celsius for the thermistor; km/h will be easier for the equation (1). Multiply the reading by 3.6 to obtain speed in km/h)

Q. Are conversion factors required, and where can they be applied?

(Yes, km/h for wind speed; and unlike metres/second can be directly used in equation (3). Simply multiply speed in metres per second by 3.6.This can be achieved via the measurement list, where a 'custom calculation' can be created (see fig 5.0).

A new variable 'windspeed km/h' is created, and equals 'speed' x 3.6, where speed contains the raw sensor reading in meters/second.)

Q. What sampling and calculation intervals are required for a usable result?

(Using five minute averages of air temperature and wind speed to calculate wind chill index was deemed appropriate for this application)

Q. What values need recording to logger memory?

(wind chill temperature, air temperature and wind speed. Although the latter two are likely logged by other functions in the weather station application)

Q. How might wind chill application break down into GP2 script

- Select sensors to be used, and create custom calculations to implement any unit conversions; e.g. from metres/second to km/h
- Script executed every minute to sum wind speed and temperature
- Script executed every five minutes to:
    1. Calculate average wind speed and temperature
    2. Apply temperature and wind speed checks to ensure wind chill conditions occurred in previous five minutes
    3. If wind chill conditions occurred, apply average readings into equation (3)
    4. Record wind chill temperature if valid
    5. Reset total temperature and total wind speed variables to 0.

Q. How can logger simulator be used to check and test script?

(Pre-set 'windspeed (km/h)' and 'temperature' variables to constant values)

Adapting the equation:

As with the dew-point example, start by breaking wind chill equation (3), in to smaller chunks replacing any common factors with constants;

$Twc = 13.12 + 0.6215Ta - 11.37V^{0.16} + 0.3965TaV^{0.16}$ (3)

$V^{0.16}$ (wind velocity raised to power of 0.16) is used twice and replaced in each case with 'pwr', where;

pwr = POW[average_w , 0.16] (in other words pwr = $average\_w^{0.16}$)

V has been replaced with average_w, and Ta with average_t, (below) as an example, uses five minute averages rather than individual readings.

eqa = 0.6215 x average_t

eqb = 11.37 x pwr

eqc = 0.3965 x average_t x pwr

twc = 13.12 + eqa – eqb + eqc (4)

Implementing this equation is more complex than the dew-point example, so the task is broken down into further small chunks, tackled separately;

1. Convert measured wind speed into km/h.
2. Calculate five minute average for both wind speed and temperature.
3. If five minute 'averages' meet criteria for wind chill event, (wind speed > 4.8 km/h and temperature <= 20°C) calculate and record wind chill index using equation (4).
4. Record average wind speed and temperatures
5. Reset averages to zero for next readings; i.e. Repeat process from **Task 2.0**

Task 1.0

A wind speed in km/h is achieved by multiplying raw reading in m/s by 3.6. Fig 5.0 shows how this is achieved from the measurement list. The result is allocated variable name windspeed (km/h).

Task 2.0

To calculate 5 minute averages for wind speed (windspeed (km/h)) and temperature (temp), a script is executed every minute to calculate running totals; i.e. total_t and total_w, (fig 7.0). The running totals are used by a second script (fig 8.0), executing every five minutes to calculate the actual average.

average_t = total_t / 5

average_w = total_w / 5

Variables are 'global' within script editor, meaning they can be used by all other scripts in that application. At this point it is also worth noting that results of calculations are affected by the order scripts are placed; (see "Testing scripts" section below)

Task 3.0

Averages are evaluated by an IF / THEN function to establish if wind chill conditions have occurred in previous five minutes; i.e. Wind speed >= 4.8 km/h & temperature <=10°C. If these conditions are met, **twc** is then calculated and recorded to logger memory using **Record (twc);**

```
IF ( average_w >= 4.8)
        AND
 ( average_t <= 10) THEN
   twc = 13.12 + eqa – eqb + eqc
   Record (twc)
END IF
```

It is worth noting **twc** is only calculated and recorded if the wind chill conditions are met, saving logger memory and reducing execution time.

Task 4.0

The Record function is also used to unconditionally log average wind speed and temperature values every five minutes, (script execution interval);

Record (average_t)

Record (average_w)

Task 5.0

'total_t' and 'total_w' are both set to zero, ready for the next measurement cycle, i.e. starting from task 2.

Total_t = 0

Total_w = 0

Testing the script(s):

Scripts can be loaded and 'testrun' by the GP2 simulator. Resulting wind speed and temperature averages can be read from output graphs and applied to equation (3). Calculated **twc** can then be compared with simulated **twc,** (fig 9.0a), they should be identical.

A flexible and more accurate approach is to pre-set average_t, and average_w to known values before simulation. These pre-set values are than applied to equation (3), as before the calculated result is compared with simulated result. The simulated **twc** should be constant and hence easier to verify with equation (see fig 9.0b)

Script timing: How do the two scripts interact?

The two scripts in this example run at different rates; i.e. 1 and 5 minute intervals. Consider 'typical' measurement cycle; (T = time)

T = 0	Wind speed and temperature totals both set to zero
T + 1 minutes	Script reads wind speed and temperature & adds results to running total
T + 2 minutes	Script reads wind speed and temperature & adds results to running total
T + 3 minutes	Script reads wind speed and temperature & adds results to running total
T + 4 minutes	Script reads wind speed and temperature & adds results to running total
T + 5 minutes	At this point both scripts are executed, with the main script (every 5 min) calculating five minute averages, and logging them with any valid wind chill temperatures (**twc**). Next measurement cycle starts at time = 0

At Time + 5, how does 'main' script correctly calculate five minute averages? i.e. ensure script for totalling wind speed and air temperature is completed before averages and **twc** are calculated.

Consider the screenshot shown in fig 6.0. Under the 'Scripts' list, the 1 minute 'totalling' routine is placed above the 5 minute averaging/wind chill calculation. Scripts are processed in list order.

For the above example, (when T + 5 minutes), temperature and wind speed are measured, and added to the running totals (total_t and total_w), before the script to calculate average and wind chill executes.

If the script order was reversed, average temperature, wind speed and wind chill, (average_t, average_w and twc) would be calculated before the fifth set of temperature and wind speed readings are made and added to running totals. Thus incorrect averages would be used to calculate **twc**.

Script order can be changed using WINDOWS 'Click & drag' function.

References:

www.wikipedia.org/wiki/Wind_chill (Last accessed: 13th May 2013)

www.mountwashington.org/weather/wind-chill.php (Last accessed: 13th May 2013)

www.erh.noaa.gov/er/iln/tables.htm (Last accessed: 13th May 2013)

www.weatheroffice.gc.ca/prods_servs/normals_documentation_e.html (Last accessed: 13th May 2013)

Fig 5.0 Creation of custom calculation from within 'measurement':

Screenshot shows 'Speed' variable ('Raw' metres/second reading) multiplied by 3.6 for conversion to Km/h. Although not shown here, the result is allocated to variable 'windspeed (km/h)'. This can also be accessed by other script functions.

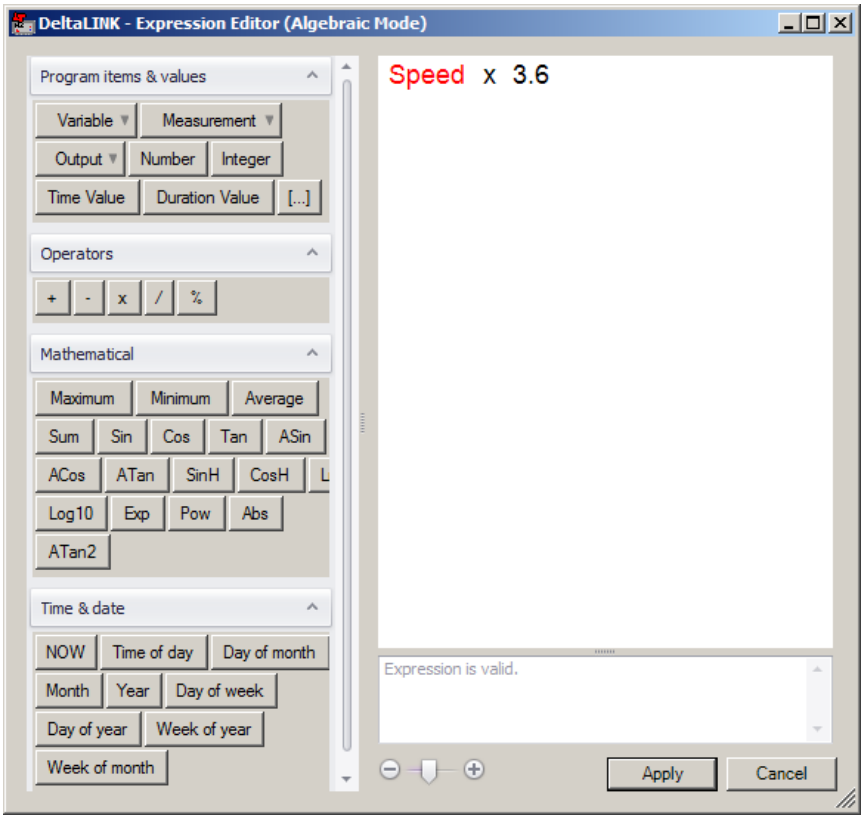


Fig 6.0 Screenshot from program editor (top level):

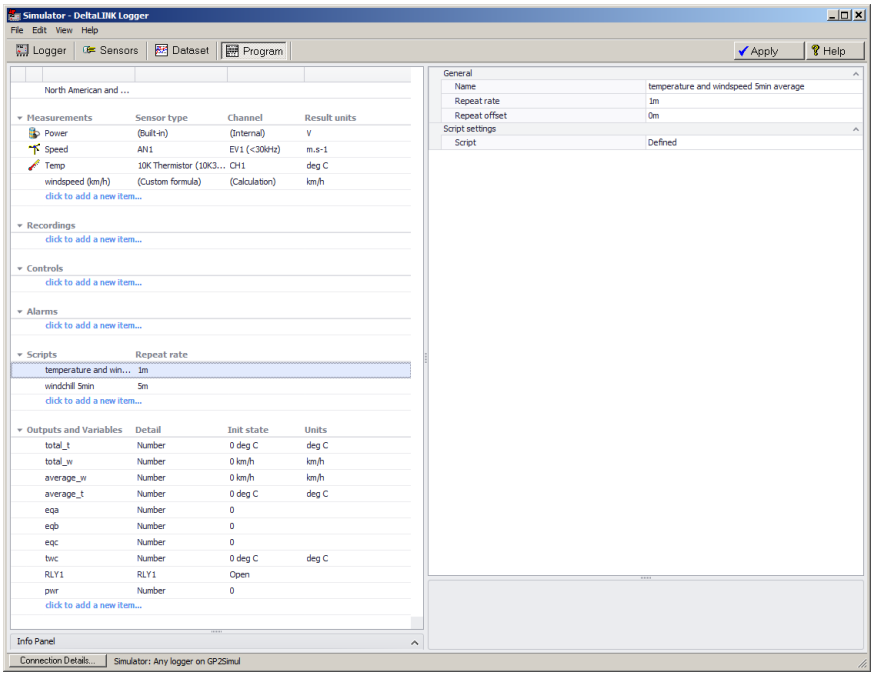


Fig 7.0 Screenshot of temperature and wind speed cumulative total(s) script:

Script executed every minute, and totals reset after every five minutes after averages calculated**.**

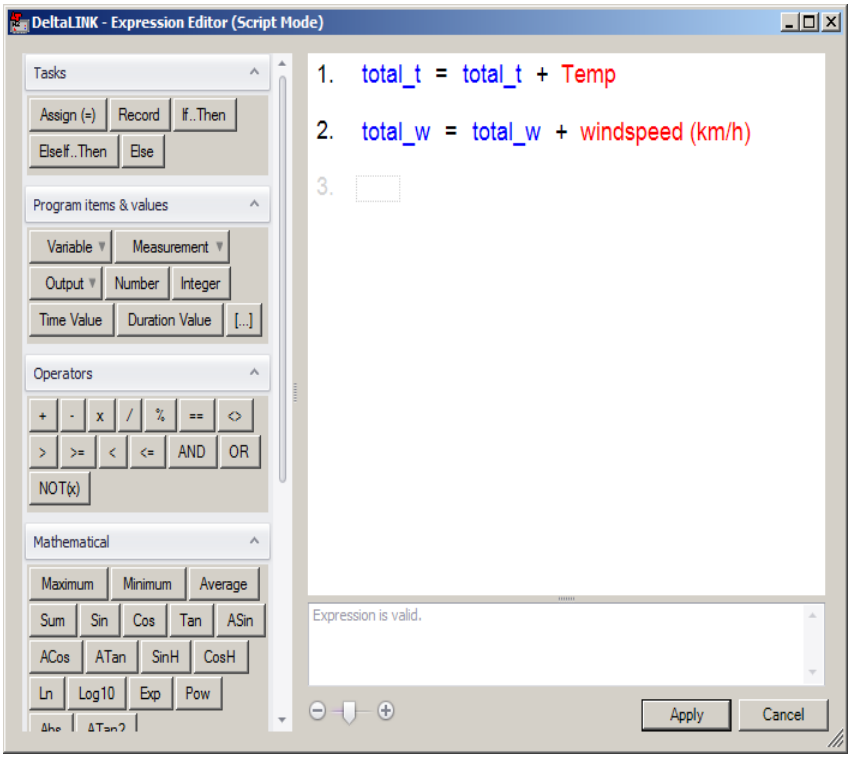


Fig 8.0 Screenshot from averaging/calculation script, executed every 5 minutes:

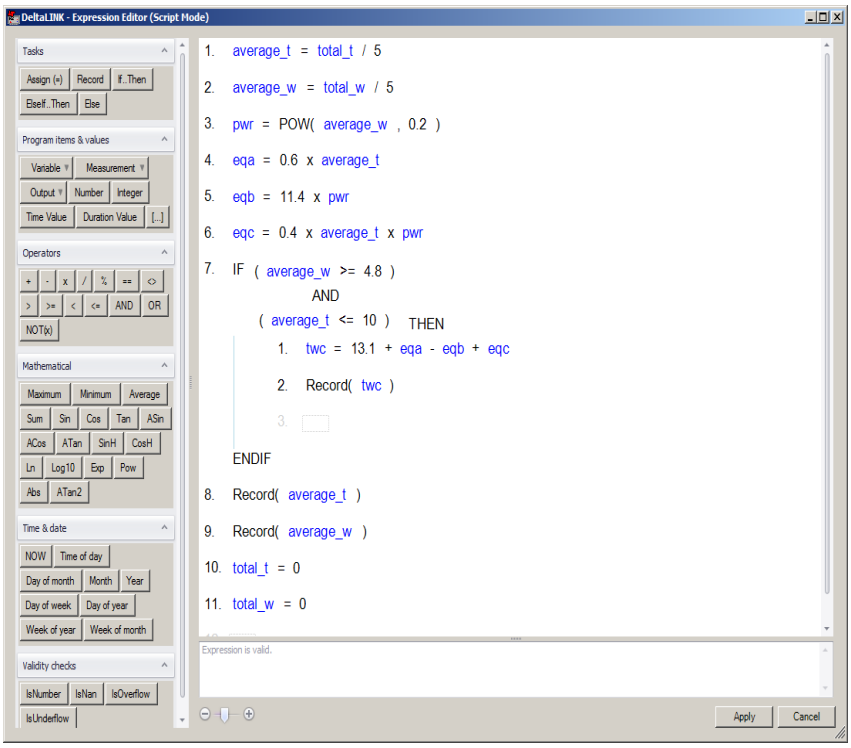


Fig 9.0 Simulation of wind chill function:

Wind chill calculated on basis of simulated temperature and wind speed.

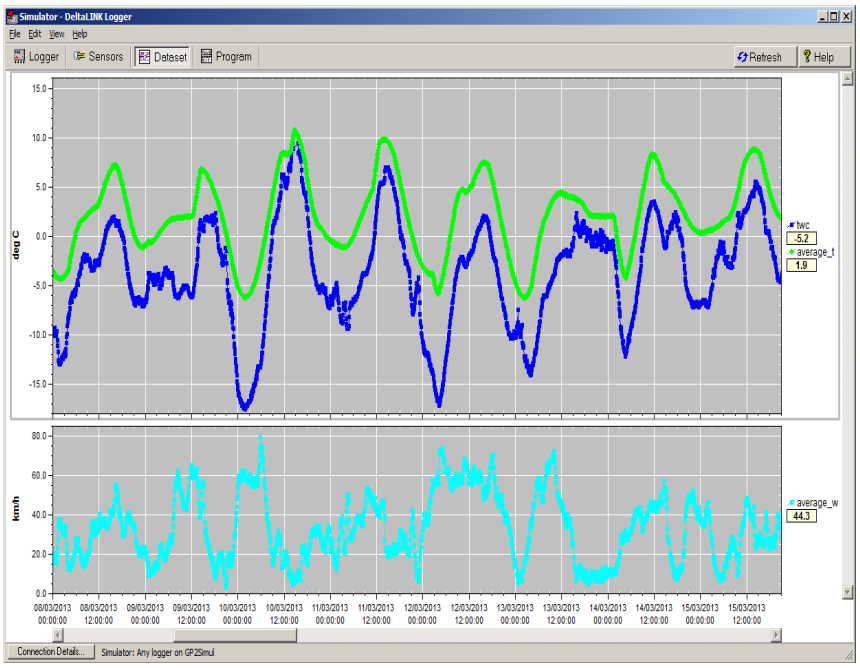
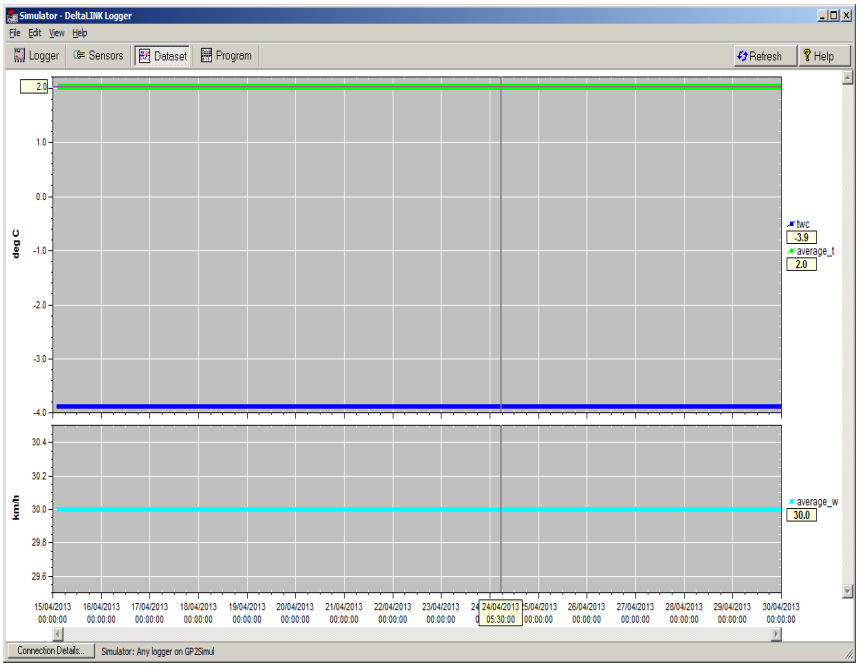


Fig 9.0b Simulation of wind chill function:

Wind chill calculated with average temperature and wind speed pre-set to 2°C and 30Km/h respectively; i.e. average_t = 2 and average_w = 30.



End

Delta-T Devices Ltd

130 Low Road,
Burwell,
Cambridge
CB25 0EJ
England

Tel: +44 (0) 1638 742922
sales@delta-t.co.uk

www.delta-t.co.uk