Accessing the IzoT Device Interface

Your application code reads and writes datapoint and property data from and to your device interface as part of its algorithm. For example, your application may sample a temperature sensor connected to physical I/O lines, process the data received, and provide the processed data to the network using an output datapoint with a **SNVT_temp_f** type, in this example implemented with an **Open Loop Sensor** block.

Example

The following example implements an **Open Loop Sensor** block with a **SNVT_temp_f** datapoint and adds the optional **nciGainproperty**.

```
#include "IzotDev.h" SFPTopenLoopSensor(sensor, SNVT_temp_f) sensor; //@IzoT block implement(nciGain)   void
sample_io(void) {    float current = get_sensor_data();
        current *= sensor.nciGain->multiplier;
        current /= sensor.nciGain->divider;
        sensor.nvoValue.data = current;
                IzotPropagate(sensor.nvoValue); }   int main(void) {     IzotInit();     while(1)
{         IzotEventPump();  your_algorithm();}     return 0; }
```

You can access block datapoint members through the member name as defined within the profile, e.g., **sensor.nvoValue**. The datapoint is implemented within the block datapoint member, and can be accessed with the data attribute, e.g., **sensor.nvoValue.data = 1234**.

Other attributes provided with each block datapoint member include the **global_index** attribute and properties which apply to the datapoint member.

You can access block property members also through the member names defined in the profile (e.g., **sensor.nciGain**). This produces a property pointer instead of a datapoint member. Properties are accessed through pointers because properties may be implemented in different ways, which can require grouping of property values separately from the block.

Even though properties can be implemented as property datapoints (configuration network variables), a property does not generally have attributes such as **global_index**.