Implementation of a weather station sensor

Olav Vassbotn

November 15, 2020

Approach

The weather sensor I wanted to implement was a simple weather station that measured the temperature, humidity and wind conditions the weather station was intended to not be a specific sensor but a generic one that would fit most sensors of this nature.

Figure 1: Simple weather station [1]

An example of a sensor that would fit the criteria can be seen in figure 1. In order to prepare the project, I researched how other sensors Dune had been implemented. This not only to help me implement the sensor but also to make it easier for anyone to get into the code. As I knew I would make use of the normal distribution I also researched how it is easiest implemented in C++. For this I want to give credit to the cplusplus.com website.

In order to test my sensor I also made a simple monitor of the sensor that would spew out any information received by the sensor. It has no other usage than for debugging purposes.

Organization

To make sure the simulated values do not conflict with other simulated values the sensor will listen for any other reported values. I did this to avoid conflicting simulated values. It could for example cause problems if there are conflicting values for the temperature. This is done purely to make it easier for users as the two completely different values of the parameters would be unrealistic and would cause unrealistic simulations. The information in the messages is not used besides this and the sensor will behave normally otherwise.

To simulate the change in the different parameters of temperature, humidity and wind I decided to implement them as Wiener processes, which is integrated white noise. This has the benefit of being a quick and easy way to get quite realistic behaving simulations without over-engineering the problem. In addition to this, I also implemented a version that also does not deviate too much from the initial value by letting the mean of the normal distribution, , be scalar of the difference between the initial value and the simulated value. This was added to let the user have an easy way to let the parameters be semi still. This can be useful if you only want quasi-random values that hover around a specific value. This has the effect of steering the Wiener process toward the 'stay on' value if it deviates a lot from the initial value.

Results

The effect of the two settings for simulation can be seen in figure 2 where the red is simulated with the stay-on effect and the blue is without. As seen with the 'stay on' effect the value of the humidity stays relatively on the targeted value.

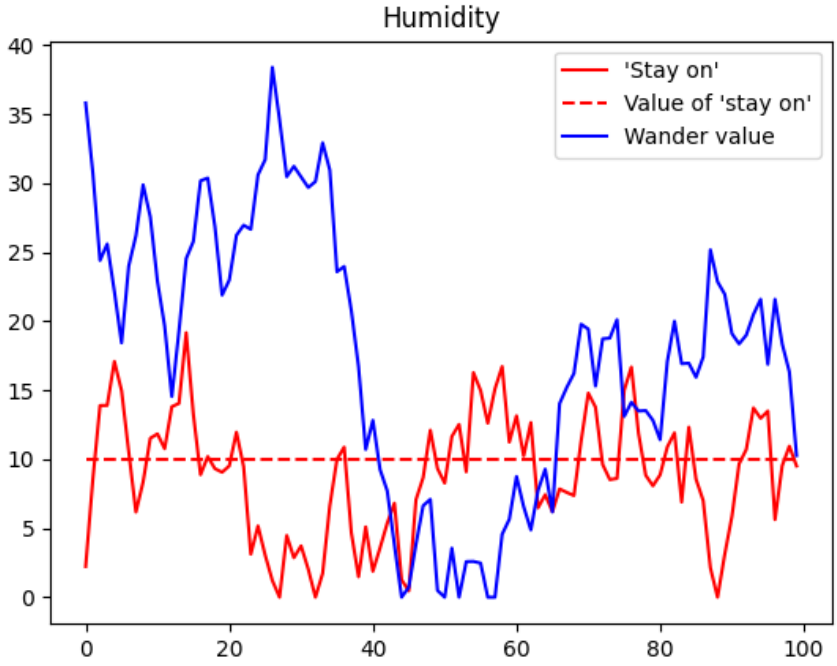

Figure 2: Section of simulated values for humidity with different settings

I tested the sensor on the lauv-xplore-1 vehicle. Though an underwater vessel has little usage for a weather station the benefits of using this vessel were that there was already a simulation of temperature, but not for any other parameter. This allowed testing of both the parameter simulation and the 'non'-conflicting parameter simulation I wanted to let the sensor have. An example of usage can be seen in figure 3.

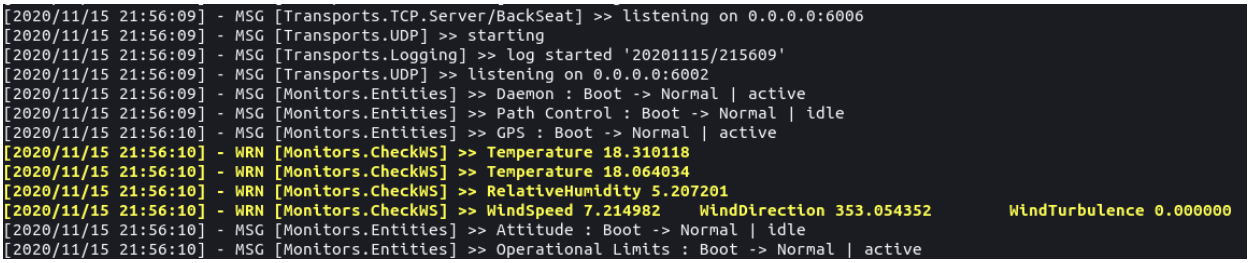

Figure 3: Output from console. Note: I used warnings only because they are easy to read and spot.

The reason there are two outputs of the temperature is that there is one module besides the weather station that transmits messages of temperature. The figure also showcases how the temperature of the module and the weather station is roughly the same besides measurement noise despite the initial temperature in the weather station was 10°.

References

[1] Amazon. Taylor weather station wind sensor from amazon. https://www.walmart.com/ip/Weather-Station-Wind-Sensor/20837325.