USING A SENQUIP DEVICE AS A DAY NIGHT CONTROLLER

1. Introduction

Day night sensors are used to automatically control the operation of lighting, solar, greenhouses and more. The output from a light sensor is compared with a setpoint to make the day night decision. Unfortunately, the sensitivity of the light sensor changes as it gets dirty and they are prone to triggering by shadows and stray light sources.

Another way to detect day and night is with a simple algorithm that uses position and time to calculate sunrise and sunset. Senquip devices are equipped with GPS and can run JavaScript locally and so can perform day night control on connected systems without a light sensor.

This application note discusses the algorithm to calculate sunrise and sunset and its implementation on a Senquip device. It is assumed that the user has Admin privileges and scripting rights for the device being worked on. To request scripting rights, contact support@senquip.com.

Before using the algorithm, it is suggested that extensive testing be performed.

2. References

The following documents were used in compiling this Application Note.

Reference	Document	Document Number
A	Sunrise Equation	Sunrise Equation, Wikipedia
B	Julian Date Converter	JD Date/Time Converter, NASA Jet Propulsion Laboratory

3. The Algorithm

The sunrise equation or sunset equation can be used to derive the time of sunrise and sunset for any location on the earth. Various implementations of the equation can be found online but after some research, it was decided to use the one presented by Wikipedia as described in Reference A.

The algorithm makes extensive use of the Julian day which is the continuous count of days since the beginning of the Julian period, and is used primarily by astronomers, and in software for easily calculating elapsed days between two events. I found it helped to think of Julian days as an ancient version of Unix time where year 1 was 4713 BC.

This simple equation converts Unix time to a Julian date:

Jdate = UNIX/86400+2440587.5

Where:

86400 is the number of seconds in a day

2440587.5 is the Julian day at 1/1/1970 0:00 UTC

The script presented in Appendix 1 has been written to reflect the order of the explanation in the Wikipedia article and variable names are similar.

For further information on algorithm, please see Reference A.

4. Testing

The algorithm was written in Port Stephens, Australia and so should reference sunrise and sunset at that location. The algorithm was tested on 17 Feb 2023. The sunrise and sunset times as indicated by www.sunrise-sunset.org are shown in Figure 1. It is interesting to note that Google disagreed by a few minutes and so did many other websites. Oddly there doesn't seem to be one number that everyone agrees on.

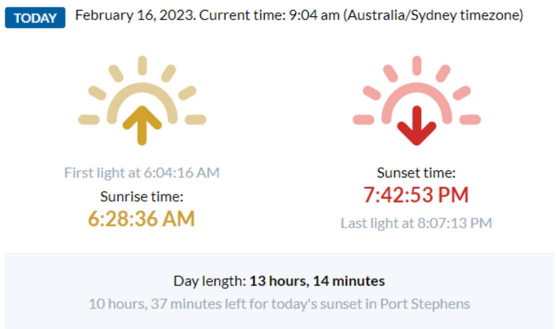

Figure 1 - Google Sunrise and Sunset

The sunrise and sunset times calculated by the algorithm are shown in Figure 2. They agree with sunrise-sunset.org and Google within a few minutes.

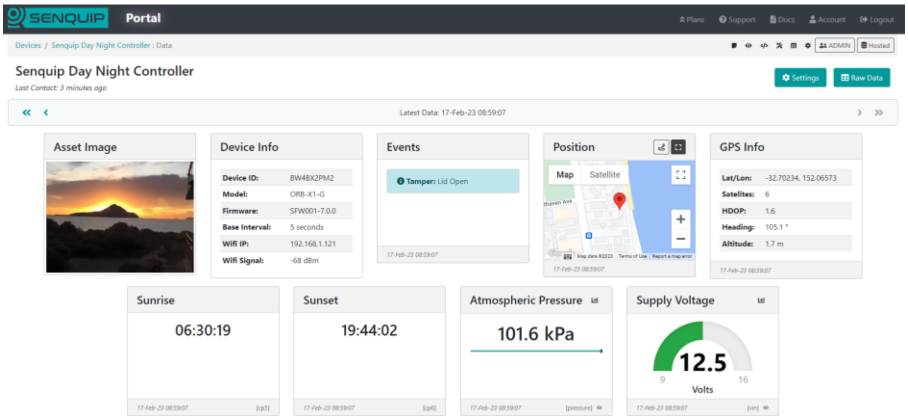

Figure 2 - Calculated Sunrise and Sunset

5. Conclusion

A simple algorithm can calculate day vs night and can be used as a no-cost replacement for light sensors.

Appendix 1: Source Code

```
load('senquip.js');
load('api_math.js');
load('api_timer.js');
load('api_config.js');

SQ.set_trigger_handler(function(tp){

 }, null);

 SQ.set_data_handler(function(data){
   let obj = JSON.parse(data);

   let timezone = 11;  // UTC ofset for Australian Eastern Daylight Saving Time
   let pi = 3.14159;
   let Jdate = Math.ceil(obj.ts/86400+2440587.5); // converting Unix time to Julian date
   let n = Jdate-2451545+0.0008;
   let Js = n-obj.gps_lon/360;
   let M = (357.5291+0.985600028*Js)%360;
   let C = 1.9148*Math.sin(pi*M/180)+0.02*Math.sin(2*pi*M/180)+0.0003*Math.sin(3*pi*M/180);
   let Lambda = (M+C+180+102.9372)%360;
   let Jtransit = 2451545+Js+0.0053*Math.sin(pi*M/180)-0.0069*Math.sin(2*pi*Lambda/180);
   let sindelta = Math.sin(pi*Lambda/180)*Math.sin(pi*23.44/180);
   let delta = 180*Math.asin(sindelta)/pi;
   let cosw0 = (Math.sin(-pi*0.83/180)-
Math.sin(pi*obj.gps_lat/180)*sindelta)/(Math.cos(pi*obj.gps_lat/180)*Math.cos(pi*delta/180));
   let w0 = 180*Math.acos(cosw0)/pi;
   let Jrise = Jtransit-w0/360;
   let Jset = Jtransit+w0/360;
```

```
    let UNIXrise = (Jrise-2440587.5)*86400;  // converting back from Julian date to Unix time
    let UNIXset = (Jset-2440587.5)*86400;

    let Localrise = Timer.fmt("%T", (UNIXrise+timezone*3600));  // using the Senquip timer library to format time
    let Localset = Timer.fmt("%T", (UNIXset+timezone*3600));

    SQ.dispatch(5,Localrise); // dispatch to the Senquip Portal
    SQ.dispatch(6,Localset);

}, null);
```