STARING A PUMP BASED ON WELL LEVEL WITH A SENQUIP DEVICE

1. Overview

This application note describes how to automatically start and stop a pump based on tank or well level using a Senquip device. The guidance applies to **any type of pump**, including:

- **Diesel-driven pumps** (with or without electronic engine controllers)
- **Solar pumps**, including variable-speed and hybrid systems
- **Electric pumps**, whether controlled by a motor drive or simple contactor
- Pumps that start via relay closure (two-wire start)

Senquip devices support all these configurations via:

- Direct output control
- **Modbus** control of smart drives or engine controllers
- **CAN/J1939** monitoring of diesel engines
- **Digital/analog feedback** inputs for pump-running, faults, flow, or temperature
- **Scripting**, enabling advanced logic such as dry-run protection, PID-style control, run-hour tracking, sequencing, and predictive starting

Two approaches are covered:

1. **Settings-Only Approach (Legacy)** – Uses the ORB's built-in warning/hysteresis/output logic.
2. **Script-Based Approach (Recommended)** – Uses a Senquip mJS script for advanced control, timing, protection, and analytics.

Although this application note references pumps, the same principles could be applied to any simple control system.

This note applies to all Senquip devices.


Figure 1 - Selection of Pumps Controlled and Monitored by Senquip Devices

For specifics on interfacing to engine controllers, see the following Senquip Application Notes:

- Modbus Integration with a Deep Sea Engine Controller
- CAN Integration with BBA Pump Controller
- Integration with Kensho Controller
- Integration with Enovation MPC-20

***Disclaimer**: The information provided in this application note is intended for informational purposes only. Users of the remote machine control system described herein should exercise caution and adhere to all relevant safety guidelines and regulations. By utilising the information provided in this application note, users acknowledge their understanding and acceptance of the associated risks. The authors and contributors disclaim any warranties, expressed or implied, regarding the accuracy or completeness of the information presented.*

2. Legacy "Settings Only" Method

3. The Legacy "Settings-Only" Method (Summary)

Earlier versions of this application note configured automation entirely through device settings. This method, although limited when compared with scripted solutions, is entirely valid and is summarised below.

Figure 2 shows a system with a 10m deep tank that is to be automatically filled to 8m when the level drops to 2m. An alarm is to be sent vis email and SMS if the water level falls below 1m or rises above 9m. A Senquip ORB is used.

A pressure sensor is used that reads 1V when the water level is 0m and 5V when the water level is 10m. Input 1 on the ORB will be used to measure the water level. The pump starts when the contacts of a relay are closed by switching the ORB output to ground. Water level measurements are to be taken every minute.

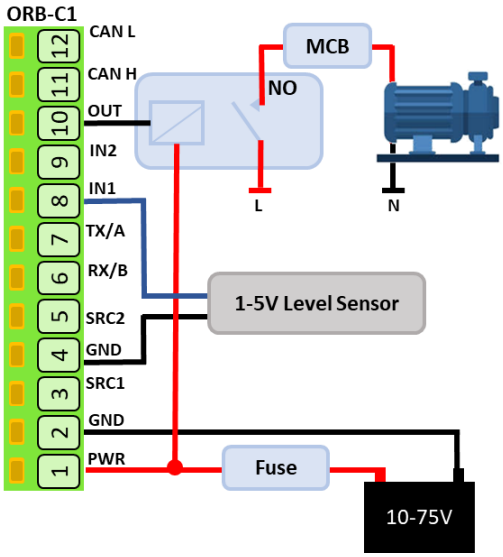

Figure 2 - Controlling Pump with Output

3.1. ORB Configuration

Only the settings applicable to this application note will be discussed. For a full description of all ORB settings, please see the ORB User Guide.

- The ORB is set to wake up every 1 minute (60 seconds), report the level and then return to sleep.
- If power is lost, the device is set to alert and then hibernate, waking every 6 hours to report in.

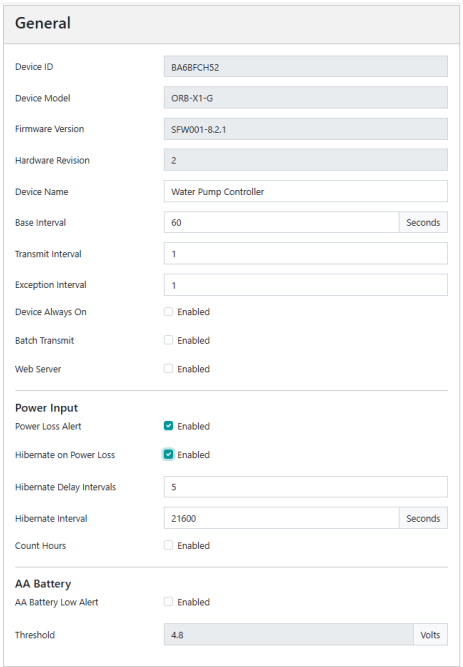

Figure 3 - General Settings

Input 1 is configured to read the sensor and apply calibration to convert the 1-5V output to a 0-10m level. In this instance a measurement of 1V represents 0m or water and a measurement of 5V represents 10m of water. A unit of "Meters" is applied to the calibrated value.

Alarms are enabled and will be used to turn on the pump when the water level reaches 2m. Hysteresis of 6m is used so that although an alarm will become active at 2m, it will only become inactive at 2m+6m = 8m, where the pump will turn off. The upper alarm level is not used and so is set to an impossible value; in this case 100m.

Warnings are enabled and are set to less than 1m or greater than 9m water level.

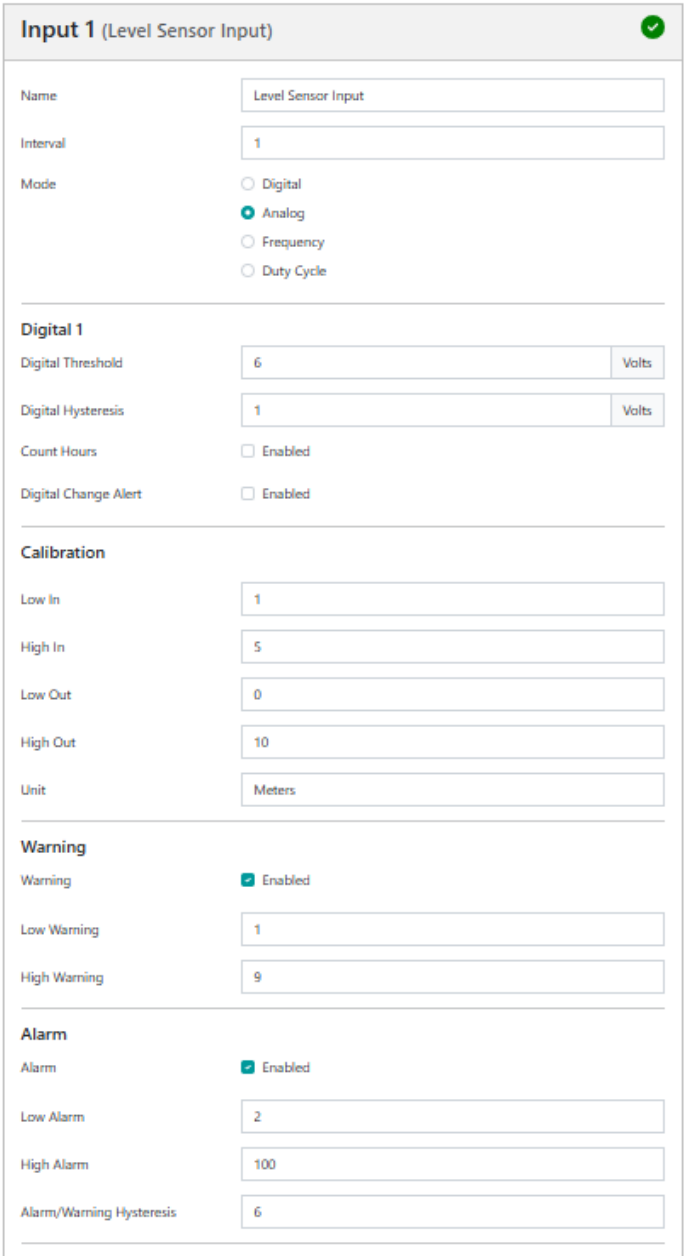

Figure 4 - Input 1 Settings

The ORB output is configured to turn on when an alarm is active. A hold time of 60 seconds is applied to ensure the pump stays on for the entire base interval of 60 seconds.

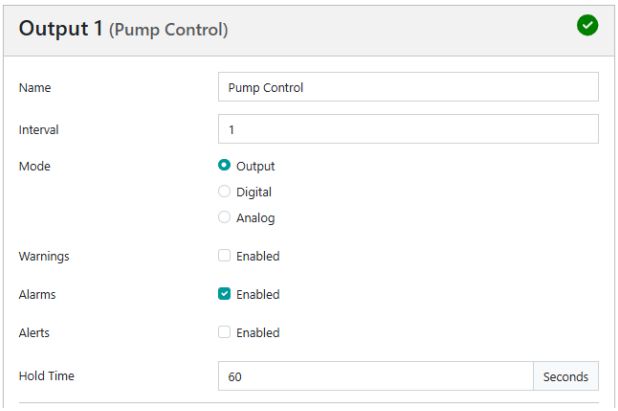

Figure 5 - Output Settings

Note: Ensure that any additional configuration options do not trigger alarms. The ORB's output is configured to turn the pump on when any alarm is active. Use alerts and warnings instead.

Event reporting via SMS and email is configured using the Events tab:

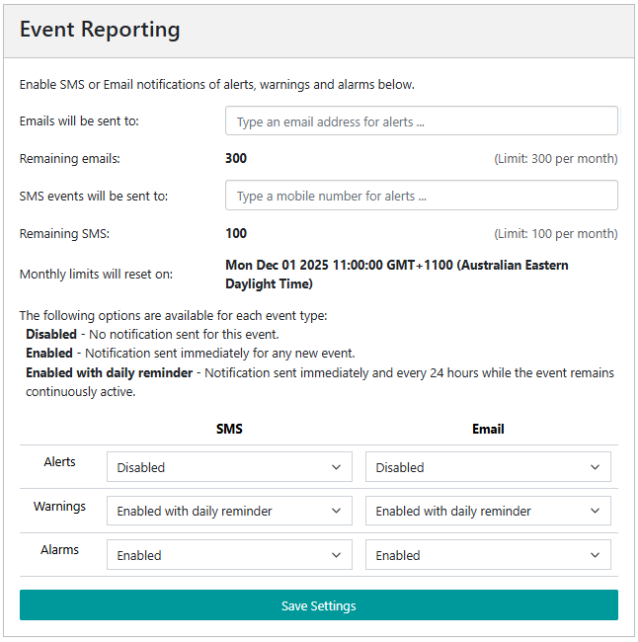

Figure 6 - SMS and Email Configuration

This approach could be improved significantly with a script to enable:

- Monitoring of the sensor to check if it goes out of allowable limits
- Checking how long the pump has been running
- Checking that the water level is increasing when the pump is on
- Monitoring the rate of change in the tank to detect leaks and theft

4. Why the Scripting Method Is Now Preferred

Senquip's scripting engine (mJS) has matured significantly. Complex automation that once required PLCs can now be done directly on a Senquip device.

Key Advantages of Using a Script

Capability	Settings-Only	Scripting (mJS)
Basic on/off control	✓	✓
Manual remote start and stop	✗	✓
Manual remote auto/manual control	✗	✓
Multiple setpoints	✗	✓
Pump run-hour tracking	✗	✓
Dry-run detection	✗	✓
Timed or staggered starts	✗	✓
PID-like proportional control	✗	✓
Adaptive behaviour (slow fill → fast fill)	✗	✓
Multiple sensors (flow + level + pressure)	Limited	✓
Logging/debugging	Limited	✓
Filtering, averaging, smoothing	✗	✓
Integration with Modbus/J1939	Basic	✓ (fully programmable)

5. Simple Start/Stop with Hysteresis (Direct Replacement of Settings Example)

This simple example executes the same functionality as the settings exampled detailed earlier. Unlike the settings method, we can easily add functionality to the script.

```
load('senquip.js');

let pump = "off";

SQ.set_data_handler(function (data) {
  let obj = JSON.parse(data);
  let level = obj.input1;

  if(level < 2){ // check the sensor level
    pump = "on";
  }
  else if (level > 8){
    pump = "off";
  }

  if (pump === "on"){SQ.set_output(1, SQ.ON, 60);} // turn output on for 60 seconds

  if (level < 1 || level > 9){SQ.dispatch_event(1,SQ.WARNING,"Level Out of Limits");}

}, null);
```

5.1. Adding Manual Control

Here we will add a script that uses trigger buttons on the Senquip Portal to allow remote start of the pump. Options for: run for 10 minutes, start, and stop are provided. Note that the automatic logic that switches the pump on at 2m and off at 8 will continue to run and will override the manual control.

```
SQ.set_trigger_handler(function(tp) {
  if (tp === 1) { SQ.set_output(1, SQ.ON, 600); }  // Turn pump on for 10 minutes
  if (tp === 2) { SQ.set_output(1, SQ.ON, 0); }    // Turn pump on
  if (tp === 3) { SQ.set_output(1, SQ.OFF, 0); }   // Turn pump off
  }, null);
```

Triggers are configured, including warning messages when the pump is to be turned on:

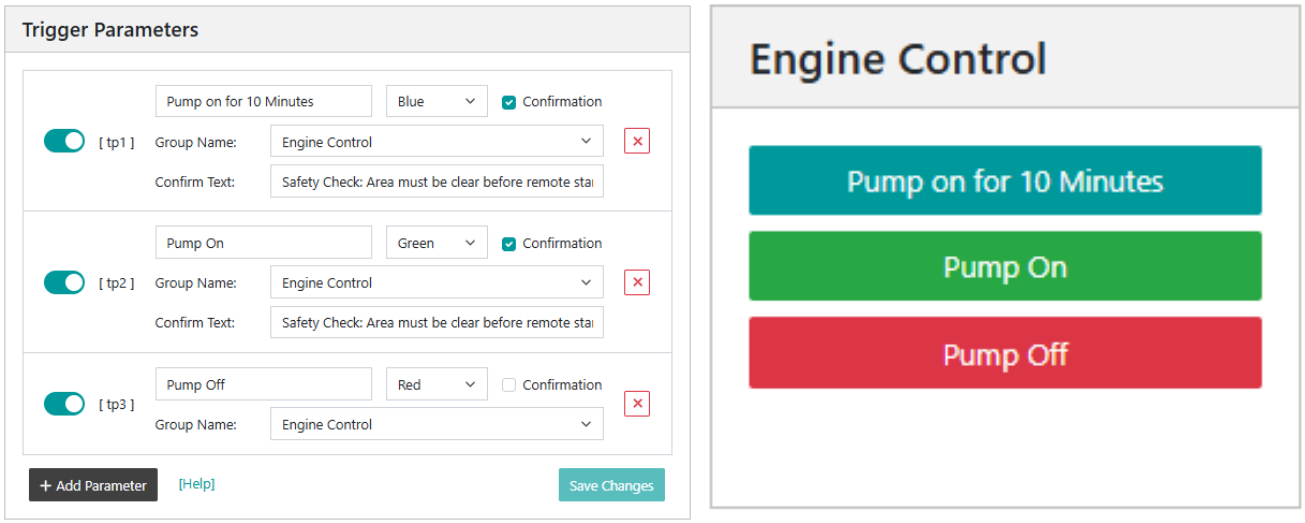

Figure 7 - Using Triggers to Enable Remote Control

5.2. Monitoring the sensor

The sensor provides 1V with no water and 5V with 10m water. With an additional *if* statement, we can generate a warning If the sensor output goes below 0.5V or above 5.5V and can stop pump operation.

Note that 0.5V from the sensor would represent -0.6m and 5.5V would represent 5.6m of water.

Sensor Voltage	Water Level
0.5	-0.625
1.0	0
5.0	5
5.5	5.625

```
load('senquip.js');

let pump = "off";

SQ.set_data_handler(function (data) {
  let obj = JSON.parse(data);
  let level = obj.input1;

  if (level < -0.6 || level > 5.6){ // check sensor operation
    SQ.dispatch_event(1,SQ.WARNING,"Sensor Fail");
    pump = "off";
  }
  else if (level < 2){ // check the sensor level
    pump = "on";
  }
  else if (level > 8){
    pump = "off";
  }

  if (pump === "on"){SQ.set_output(1, SQ.ON, 60);} // turn output on for 60 seconds

  if (level < 1 || level > 9){SQ.dispatch_event(1,SQ.WARNING,"Level Out of Limits");}

}, null);
```

5.3. Dry-Run / No Flow Protection

We can monitor the rise in tank level after a time to ensure that the pump is pumping at the expected rate.

In this example, we will assume that the Senquip device is not going to sleep and that the device is awake when the rise in tank level is calculated. A simple counter accumulates 60 second intervals, and if the pump remains on for all 10 cycles, the level is checked to ensure it has risen by the expected amount.

Only the snippet that does the level rise check is shown. At this point, the script is getting messy and could do with a re-write with all the application requirements known in advance. A simplified script is provided in Appendix A.

```
if (pump === "on"){
    SQ.set_output(1, SQ.ON, 60);  // turn output on for 60 seconds
    if (cycles === 0){
      start_level = level; // save the curent tank level
    }
    cycles = cycles + 1;
    if (cycles >= 10){
      if (level-start_level < 0.5){ // if the tank has not gone up by 0.5m
        SQ.dispatch_event(1,SQ.WARNING,"Pump Fail");
        pump = "off";
      }
      cycles = 0; // reset the counter to check again in 10 minutes
    }
  }
  else{
    cycles = 0; // clear the cycle counter;
  }
```

6. Conclusions

The original settings-only method remains a simple and proven approach for basic on/off control. However, modern Senquip scripting provides:

- Better safety
- Better performance
- Better automation
- Better diagnostic visibility
- Customer-specific logic without firmware changes

This makes scripting the preferred method for real-world pump automation, especially for mines, irrigation, bores, tanks, and remote sites.

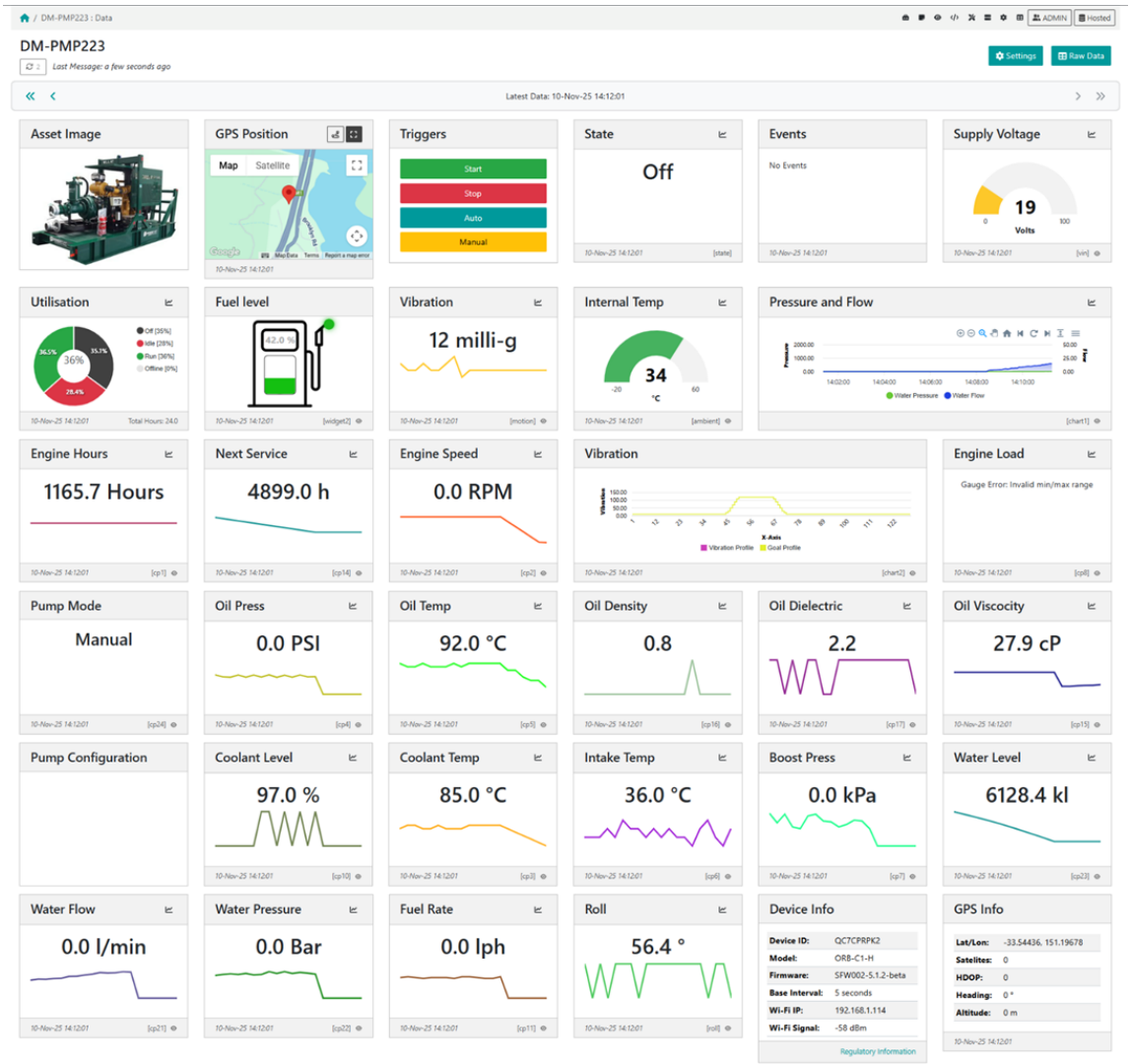

Figure 8 - Typical Pump Display on the Senquip Portal

Appendix A – Full Pump Control Script

```
load('senquip.js');

// ------------------------------------------------
// Configuration (adjust these to suit the application)
// ------------------------------------------------

// Valid calibrated level range (m) - used to detect sensor failure
let SENSOR_MIN_LEVEL   = -0.6;    // anything below this is impossible
let SENSOR_MAX_LEVEL   =  5.6;    // anything above this is impossible

// Automatic pump on/off thresholds (m)
let START_LEVEL        = 2.0;     // start pump when level falls below this
let STOP_LEVEL         = 8.0;     // stop pump when level rises above this

// Level warning limits (m)
let LEVEL_WARN_LOW     = 1.0;     // warning if below this
let LEVEL_WARN_HIGH    = 9.0;     // warning if above this

// Dry-run (no-flow) detection
let DRY_RUN_CYCLES     = 10;      // number of 60 s cycles to check (10 min)
let DRY_RUN_MIN_RISE   = 0.5;     // minimum expected rise in level over DRY_RUN_CYCLES (m)

// ------------------------------------------------
// State variables
// ------------------------------------------------

let pump        = "off";  // "on" or "off" (automatic control)
let cycles      = 0;      // counts how many 60 s periods the pump has been on
let start_level = 0;      // level at start of the dry-run check window

// ------------------------------------------------
// Main data handler
// ------------------------------------------------
// Called whenever new data is available. Assumes input1 is the
// calibrated tank level in metres (using ORB/QUAD calibration).
// ------------------------------------------------

SQ.set_data_handler(function (data) {

  let obj   = JSON.parse(data);
  let level = obj.input1;     // calibrated tank level (m)

  // ----------------------------
  // 1. Sensor sanity check
  // ----------------------------
  if (level < SENSOR_MIN_LEVEL || level > SENSOR_MAX_LEVEL) {
    SQ.dispatch_event(1, SQ.WARNING, "Sensor Fail");
    pump   = "off";   // force pump off if sensor looks bad
    cycles = 0;       // reset dry-run counter
  }

  // ---------------------------------------
  // 2. Automatic start/stop with hysteresis
  // (only if sensor is OK)
  // ---------------------------------------
  else {

    // Start pump when level is below START_LEVEL
    if (level < START_LEVEL) {
      pump = "on";
    }
    // Stop pump when level is above STOP LEVEL
    else if (level > STOP_LEVEL) {
      pump = "off";
```

```
    }
  }

  // ---------------------------------------
  // 3. Dry-run / no-flow protection
  // ---------------------------------------
  if (pump === "on") {

    // Keep the pump output on for the next 60 seconds.
    // This call is refreshed every time the handler runs.
    SQ.set_output(1, SQ.ON, 60);

    // First cycle with pump on: capture starting level
    if (cycles === 0) {
      start_level = level;
    }

    // Count how many 60 s periods the pump has been on
    cycles = cycles + 1;

    // After DRY_RUN_CYCLES periods, check that level has increased
    if (cycles >= DRY_RUN_CYCLES) {

      if ((level - start_level) < DRY_RUN_MIN_RISE) {
        // Level has not risen enough - likely dry-run or pump fault
        SQ.dispatch_event(1, SQ.WARNING, "Pump Fail (Dry Run)");
        pump = "off";   // stop automatic pumping
      }

      // Reset counter for next window
      cycles = 0;
    }
  }
  else {
    // Pump is off: always reset the dry-run counter
    cycles = 0;
    // Optionally ensure output is off (not strictly necessary if
    // the output is only ever driven from this script).
    SQ.set_output(1, SQ.OFF, 0);
  }

  // ---------------------------------------
  // 4. Level out-of-limits warning
  // ---------------------------------------
  if (level < LEVEL_WARN_LOW || level > LEVEL_WARN_HIGH) {
    SQ.dispatch_event(1, SQ.WARNING, "Level Out of Limits");
  }

}, null);

// ------------------------------------------------------
// Trigger handler - manual remote control from portal
// ------------------------------------------------------
// Configure three trigger buttons on the Senquip Portal:
//
//  Trigger 1: "Run 10 minutes"
//  Trigger 2: "Start"
//  Trigger 3: "Stop"
// ------------------------------------------------------

SQ.set_trigger_handler(function (tp) {

  if (tp === 1) {
    // Turn pump on for 10 minutes (600 seconds)
    SQ.set_output(1, SQ.ON, 600);
  }
```

```
  if (tp === 2) {
    // Turn pump on indefinitely (until script or another trigger turns it off)
    SQ.set_output(1, SQ.ON, 0);
  }

  if (tp === 3) {
    // Turn pump off
    SQ.set_output(1, SQ.OFF, 0);
  }

}, null);
```