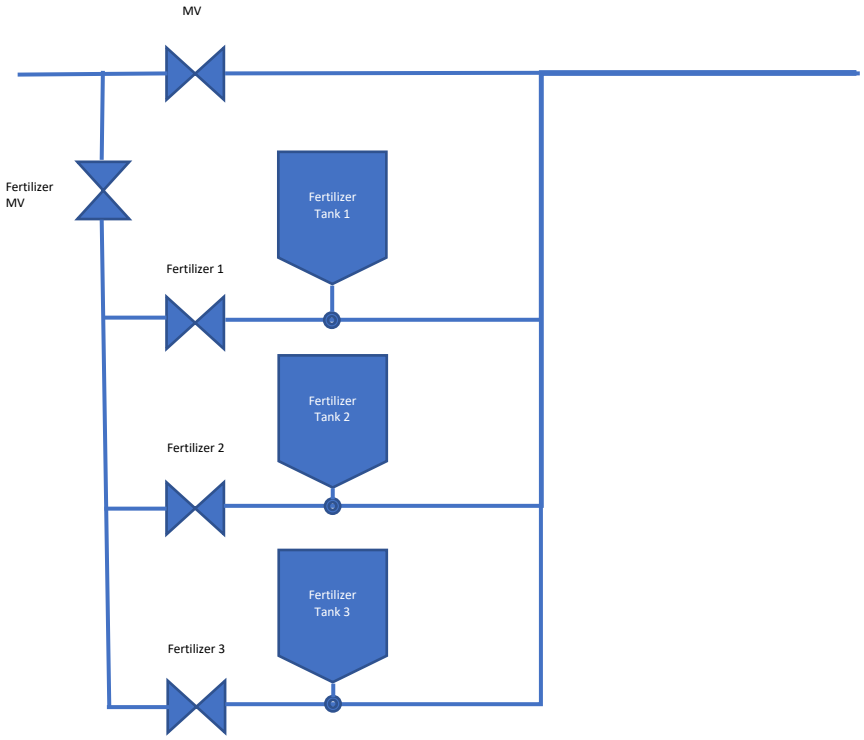
Topic: Fertilizer Control with 10 fertilizers and filter flush
Author: Søren Haxvig
Date: 2024.01.03. Rev 2: 2024.03.22
To: OEM Partners
Purpose: Demonstrate how to use the logic engine to control fertilizer injection combined with filter flush.
FW: Minimum 2.08

Principle:

Fertigation:

The fertigation control described in this document is based on the following setup:



The principle is that the water runs through the MV or through one or more fertilizer pipes, but never through the MV and fertilizers at the same time.

The flow rule setup:

FLOW VALVE FLOW SENSOR

	Id	Active	Type	State	Priority	IO	Flow sensor	Start delay (sec)	Stop delay (sec)	Capacity (l/m)	Error	Error code
	4	✓	Normally closed	Closed	0	MAIN MV		0	0	1201.00	None	0
	5		Normally closed	Closed	0	FERTILIZER VIRTUAL MV		0	0	0.10	None	0

Filter flush:

The filter flush is based on a differential pressure between in inlet and outlet of the filter triggering a filter flush if it becomes above a certain level. Once triggered the irrigation will be paused and the flushing runs a user defined time whereafter it will resume irrigation.

Control of fertilizer activation:

The logic engine is used to control the flow through the MV or the fertigation pipes in combination with the hydraulic setup. The hydraulic system comprises two MVs in parallel where one of them is the MV in the setup above and the other is a virtual fertilizer MV for all the fertigation pipes. This fertilizer MV is linked to a line unit/IO, which must be created as a MK2 decoder. The logic engine will switch the hydraulic setup between using the MV or the fertilizer MV depending on whether there is a need for fertigation. When using fertigation, it adjusts the capacity of the fertilizer MV in accordance with the requested fertilizers and turn on these accordingly.

Variables:

Id	Description	State	Type	Source id	Source sub id	Value
1	Request for fertilizers	Ok	Local	0	0	0
2	Fertilizer 1 capacity	Ok	Local	0	0	1000
3	Fertilizer 1 address	Ok	Local	0	0	12345
4	Fertilizer 1 index	Ok	Local	0	0	1
5	Fertilizer 2 capacity	Ok	Local	0	0	2000
6	Fertilizer 2 address	Ok	Local	0	0	12345
7	Fertilizer 2 index	Ok	Local	0	0	2
8	Fertilizer 3 capacity	Ok	Local	0	0	3000
9	Fertilizer 3 address	Ok	Local	0	0	12345
10	Fertilizer 3 index	Ok	Local	0	0	3
11	Fertilizer 4 capacity	Ok	Local	0	0	4000
....						
28	Fertilizer 9 index	Ok	Local	0	0	3
29	Fertilizer 10 capacity	Ok	Local	0	0	10000
30	Fertilizer 10 address	Ok	Local	0	0	23456
31	Fertilizer 10 index	Ok	Local	0	0	4
32	Last fertilizer setting	Ok	Local	0	0	0
33	Fertilizer flow rate	Ok	Local	0	0	1
34	MV Id	Ok	Local	0	0	4
35	Fertilizer dummy Id	Ok	Local	0	0	5
36	State of fertilizer MV	Ok	We flow valve state	5	0	1
37	Fertilizers active	Ok	Local	0	0	0

Variable #1 is the requested fertilizer in a bit format. 0 means no fertilizer. 1 means fertilizer #1, 2 means fertilizer #2, 4 means fertilizer #3, 5 means fertilizer #1+3 etc. This is the dynamic control variable to turn on/off fertilizers. The maximum is 0x3FF which will turn on all 10 fertilizers.
Variable #2 is the capacity of the pipe for fertilizer #1.
Variable #3 is address of the decoder for fertilizer #1.
Variable #4 is the output number of the decoder for fertilizer #1.
Variable #5-7, #8-10, #11-13, #14-16, #17-19, #20-22, #23-25, #26-28, and #29-31 are as above for fertilizer #2 to #10.
Variable #32 is for internal use.
Variable #33 is the actual flow capacity for the active fertilizers.

Variable #34 is the ID for the MV on the flow rule.
Variable #35 is the ID for the fertilizer MV on the flow rule.
Variable #36 is the state of the fertilizer MV.
Variable #37 is for internal use.

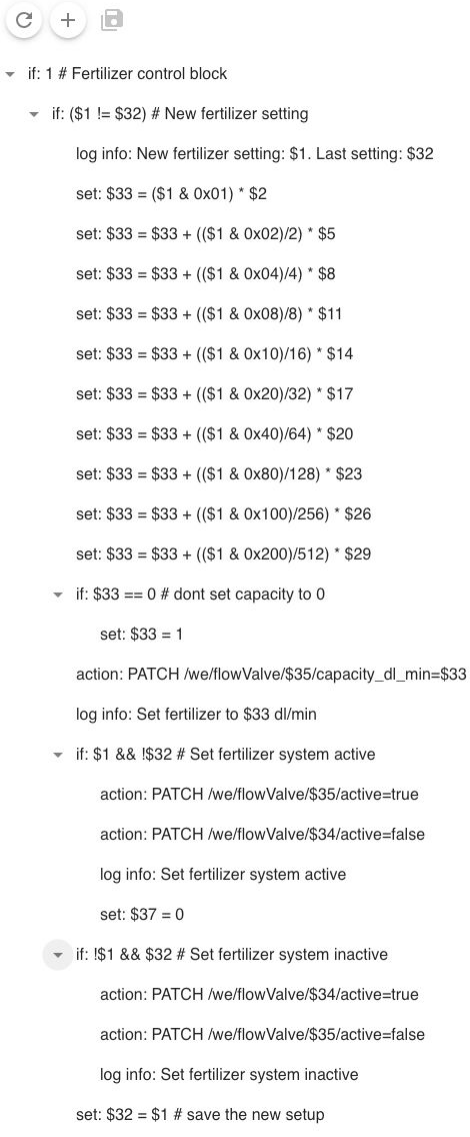
The fertilizer control is made with a variable for easy modifications on the variable page or via the API. Using the API:

```
PATCH /le/var/1/value=5
{"result":"ok"}
Means turn on fertilizer #1 and #3. Set the value=0 means turn
off fertilizers and revert to MV.
```

The other non-internal variables could be embedded in the block code, but it is more convenient to set them via variables.

Block:

Changing the request for fertilizers part:



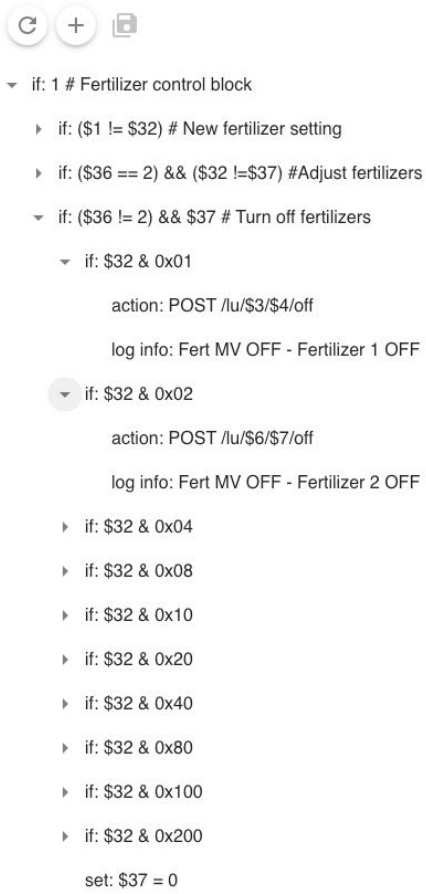
The first half calculates the flow capacity of the selected fertilizers and set it for the virtual fertilizer MV.
The next part sets the fertilizer system active by swapping active state between the MV and virtual fertilizer MV.
The last part swap back to the MV when fertilizers are no longer requested.

Turning ON fertilizers:

- if: 1 # Fertilizer control block
  - if: ($1 != $32) # New fertilizer setting
  - if: ($36 == 2) && ($32 !=$37) #Adjust fertilizers
    - if: ($32 & 0x01) && !($37 & 0x01)
      - action: POST /lu/$3/$4/on
      - log info: Irr ON - Fertilizer 1 ON
    - if: !($32 & 0x01) && ($37 & 0x01)
    - if: ($32 & 0x02) && !($37 & 0x02)
    - if: !($32 & 0x02) && ($37 & 0x02)
      - action: POST /lu/$6/$7/off
      - log info: Irr ON - Fertilizer 2 OFF
    - if: ($32 & 0x04) && !($37 & 0x04)
    - if: !($32 & 0x04) && ($37 & 0x04)
    - if: ($32 & 0x08) && !($37 & 0x08)
    - if: !($32 & 0x08) && ($37 & 0x08)
    - if: ($32 & 0x10) && !($37 & 0x10)
    - if: !($32 & 0x10) && ($37 & 0x10)
    - if: ($32 & 0x20) && !($37 & 0x20)
    - if: !($32 & 0x20) && ($37 & 0x20)
    - if: ($32 & 0x40) && !($37 & 0x40)
    - if: !($32 & 0x40) && ($37 & 0x40)
    - if: ($32 & 0x80) && !($37 & 0x80)
    - if: !($32 & 0x80) && ($37 & 0x80)
    - if: ($32 & 0x100) && !($37 & 0x100)
    - if: !($32 & 0x100) && ($37 & 0x100)
    - if: ($32 & 0x200) && !($37 & 0x200)
    - if: !($32 & 0x200) && ($37 & 0x200)
    - set: $37 = $32

The logic above awaits on the top level IF state of the virtual fertilizer MV ($36) is ON and there has been a change in the settings ($32 != $37). Turning ON is controlled by the flow management when there is a need for water.
The next two IF statements turn ON or OFF the fertilizer valve 1 depending on the request. The next nine sets if IF do the same for fertilizer valve 2 to 10. On for fertilizer 1 and Off for fertilizer 2 are unfolded in the example above.

Turning OFF fertilizers when irrigation stops:



This section will stop any active fertilizer valves when the fertilizer MV stops.

How to operate:

As stated above there are some initial setup of the capacity, identification of the fertilizer valves etc which can be done via API calls. For all of them it is a matter of setting the “value” except for variable 34 where it is the “sourceId” which must be set. All this is once for an installation.

After the setup the activation of fertilizers is just to control the variable #1. The logic handles both that the fertilizers are set in advanced and if they are applied during irrigation. The setting can also be changed during irrigation. This means an irrigation can be started on the MV without any fertilizers and then after some time the fertilizers are enabled. During the run they can be adjusted and finally stopped while the irrigation continues to clean the pipes.

Example:

Timestamp	Severity	Type	Data
Fri, 01 Sep 2023 15:50:25	Info	we::station::stop	id: 17, water_p: 0, emitted_dl: 466, runtime_se
Fri, 01 Sep 2023 15:50:23	Info	we::flowValve::close	id: 4, ioAddr: 676088648, ioIdx: 6
Fri, 01 Sep 2023 15:50:03	Info	we::flowValve::open	id: 4, ioAddr: 676088648, ioIdx: 6
Fri, 01 Sep 2023 15:49:58	Info	le::user::log	logStr: Irr ON - Fertilizer 3 OFF
Fri, 01 Sep 2023 15:49:54	Info	we::flowValve::close	id: 5, ioAddr: 1001, ioIdx: 1
Fri, 01 Sep 2023 15:49:49	Info	le::user::log	logStr: Irr ON - Fertilizer 2 OFF
Fri, 01 Sep 2023 15:49:47	Info	le::user::log	logStr: Set fertilizer system inactive
Fri, 01 Sep 2023 15:49:47	Info	le::user::log	logStr: Set fertilizer to 1 dl/min
Fri, 01 Sep 2023 15:49:47	Info	le::user::log	logStr: New fertilizer setting: 0
Fri, 01 Sep 2023 15:49:22	Info	le::user::log	logStr: Irr ON - Fertilizer 3 ON
Fri, 01 Sep 2023 15:49:16	Info	le::user::log	logStr: Irr ON - Fertilizer 1 OFF
Fri, 01 Sep 2023 15:49:14	Info	le::user::log	logStr: Set fertilizer to 10000 dl/min
Fri, 01 Sep 2023 15:49:14	Info	le::user::log	logStr: New fertilizer setting: 6
Fri, 01 Sep 2023 15:48:46	Info	le::user::log	logStr: Irr ON - Fertilizer 2 ON
Fri, 01 Sep 2023 15:48:40	Info	le::user::log	logStr: Irr ON - Fertilizer 1 ON
Fri, 01 Sep 2023 15:48:35	Info	we::flowValve::open	id: 5, ioAddr: 1001, ioIdx: 1
Fri, 01 Sep 2023 15:48:31	Info	we::flowValve::close	id: 4, ioAddr: 676088648, ioIdx: 6
Fri, 01 Sep 2023 15:48:29	Info	le::user::log	logStr: Set fertilizer system active
Fri, 01 Sep 2023 15:48:29	Info	le::user::log	logStr: Set fertilizer to 6000 dl/min
Fri, 01 Sep 2023 15:48:29	Info	le::user::log	logStr: New fertilizer setting: 3
Fri, 01 Sep 2023 15:48:05	Info	we::station::start	id: 17, ioAddr: 676088648, ioIdx: 3, water_p: 0
Fri, 01 Sep 2023 15:48:03	Info	we::flowValve::open	id: 4, ioAddr: 676088648, ioIdx: 6

In the example above a station # 17 is started downstream the MV. This makes MV #4 start (flowValve). Then fertilizers are requested for 1 and 2 (bitwise 3). MV is swapped to fertilizer MV #5 and the fertilizer valves are opened. Later the request is changed to 2+3 (bitwise 6) and fertilizer 3 is opened, #1 is closed and #2 untouched. Finally, the request is set to none and fertilizer #2 and #3 are closed.

Control of filter flush with one sensor input:

This setup is based on one sensor decoder reading a differential pressure on the filter.

Variables:

	Id	Description	State	Type	Source id	Source sub id	Value
✎	38	Two wire state	Ok	Tw state	0	0	2
✎	39	Diff. Pressure Transducer value	Ok	IO value	679088867	1	4400
✎	40	Flush valve value	Ok	IO value	23456	6	0
✎	41	Timer	Ok	Local	0	0	0
✎	42	Trigger level for act. (mBar)	Ok	Local	0	0	5000
✎	43	Flush time in seconds	Ok	Local	0	0	100
✎	44	Stable time for trigger in sec	Ok	Local	0	0	60
✎	45	Flush valve address	Ok	Local	0	0	23456
✎	46	Flush valve index	Ok	Local	0	0	6

Variable #38 is the state of the 2-wire. It is important to make sure the 2-wire is operational while reading and reacting on sensors.
Variable #39 is the input from a sensor decoder. In this example the conversion via the IO configuration is made to convert the 4-20mA signal into mBar (See conversion setup at the end of this document).
Variable #40 is the output decoder controlling the flush valve.
Variable #41 is a timer used to make sure the sensor reading is stable before doing any actions.
Variable #42 is the trigger level (above this) for activating the filter flush.
Variable #43 is filter flush time in seconds.
Variable #44 is the stable time for the differential pressure before activating the filter flush.
Variable #45 is the address of the filter flush valve. Must be set to the same as Source id in #40.
Variable #46 is the index of the filter flush valve. Must be set to the same as Source sub id in #40.

Block:

```
if: $38 == 2 && (($39 && ($39 > $42)) || $40)
    set: $41 = $44 # Set stable count
    while: $39 && ($39 > $42) && !$40 && $41
        set: $41 = $41 - 1
        sleep: 1
    if: $40 && !$41
        log info: Flush valve manually opened
    if: !$40 && !$41
        action: PATCH /we/engine/targetState=3 # Pause engine
        sleep: 10
        action: POST /lu/$45/$46/on # Open flush valve
        log info: Flush valve opened at $39 mBar
    sleep: $43 # flush in progress
    if: !$39
        log warning: Unable to read level from pressure transducer
    if: $38 != 2
        log info: Two-wire disabled
    if: $38 == 2 && !$40
        log info: Flush valve manually closed
    if: $40 && !$41
        action: POST /lu/$45/$46/off # Close flush valve
        action: PATCH /we/engine/targetState=2 # Resume engine
        log info: Flush valve closed at $39 mBar
```

The first “if” statement checks that the 2-wire is operational, and the pressure is above the limit, or the flush valve is already open.
The while blocks checks that the pressure stays above the limit for the stable period.
The 3rd “if” block starts setting the irrigation engine in pause, wait for that in 10 seconds and then open the flush valve.
Wait the defined flush time seconds for flushing before it continues.
Last “if” block closes the flush valve and set the irrigation engine back to auto mode.

Individual pressure input on filter inlet and outlet:

This setup is based on two sensor decoders reading a differential pressure on the filter.

Variables:

Add two more variables to the setup above for a single sensor input and redefine the previous input as:

Id	Description	State	Type	Source id	Source sub id	Value
38	Two wire state	Ok	Tw state	0	0	2
39	Diff. Pressure Transducer value	Ok	Local	0	0	5645
40	Flush valve value	Ok	IO value	23456	6	0
41	Timer	Ok	Local	0	0	58
42	Trigger level for act. (mBar)	Ok	Local	0	0	5000
43	Flush time in seconds	Ok	Local	0	0	100
44	Stable time for trigger in sec	Ok	Local	0	0	60
45	Flush valve address	Ok	Local	0	0	23456
46	Flush valve index	Ok	Local	0	0	6
47	Filter In	Ok	IO value	679088867	1	6050
48	Filter Out	Ok	IO value	679088846	1	426

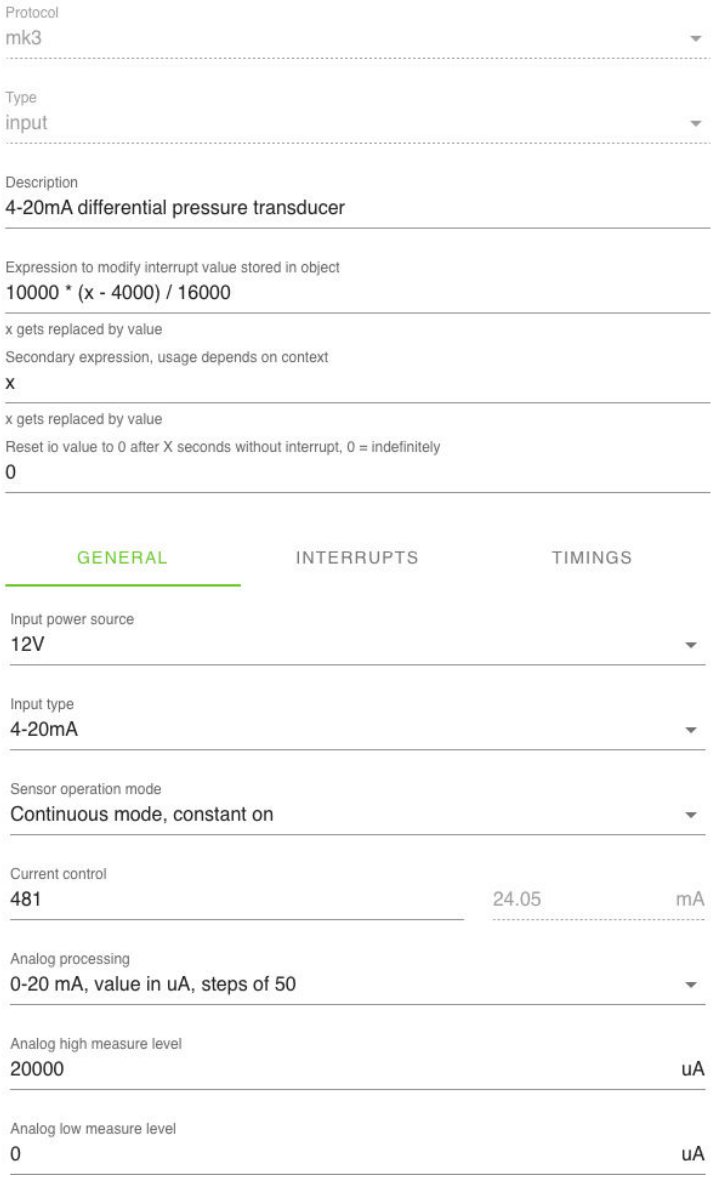
Variable #39 is changed to a local variable.
Variables #47 & #48 are sensor decoder input for inlet and outlet.
All other variables remain the same.

Block:

```
if: $38 == 2 && (($39 && ($39 > $42)) || $40)
    set: $41 = $44 # Set stable count
    while: $39 && ($39 > $42) && !$40 && $41
        set: $41 = $41 - 1
        sleep: 1
    if: $40 && !$41
        log info: Flush valve manually opened
    if: !$40 && !$41
        action: PATCH /we/engine/targetState=3 # Pause engine
        sleep: 10
        action: POST /lu/$45/$46/on # Open flush valve
        log info: Flush valve opened at $39 mBar
    sleep: $43 # flush in progress
    if: !$39
        log warning: Unable to read level from pressure transducer
    if: $38 != 2
        log info: Two-wire disabled
    if: $38 == 2 && !$40
        log info: Flush valve manually closed
    if: $40 && !$41
        action: POST /lu/$45/$46/off # Close flush valve
        action: PATCH /we/engine/targetState=2 # Resume engine
        log info: Flush valve closed at $39 mBar
if: $47 && $48
    set: $39 = $47 - $48
```

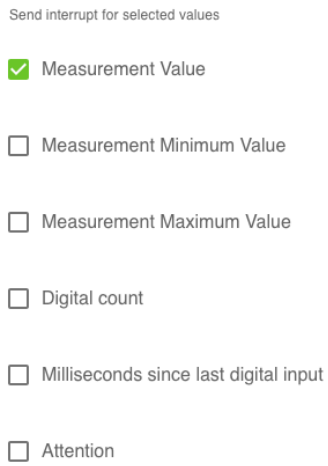
Sensor decoder setup:

IO configuration:



The expression converts the 4000-20000uA to 0-10000mBar.
The Input power source of 12V might differ depending on the actual sensor.

The setting above will update the value every 5 seconds regardless of there is any change. A minimum uA change could be defined to avoid too many interrupts on the 2-wire if the sensor fluctuates.

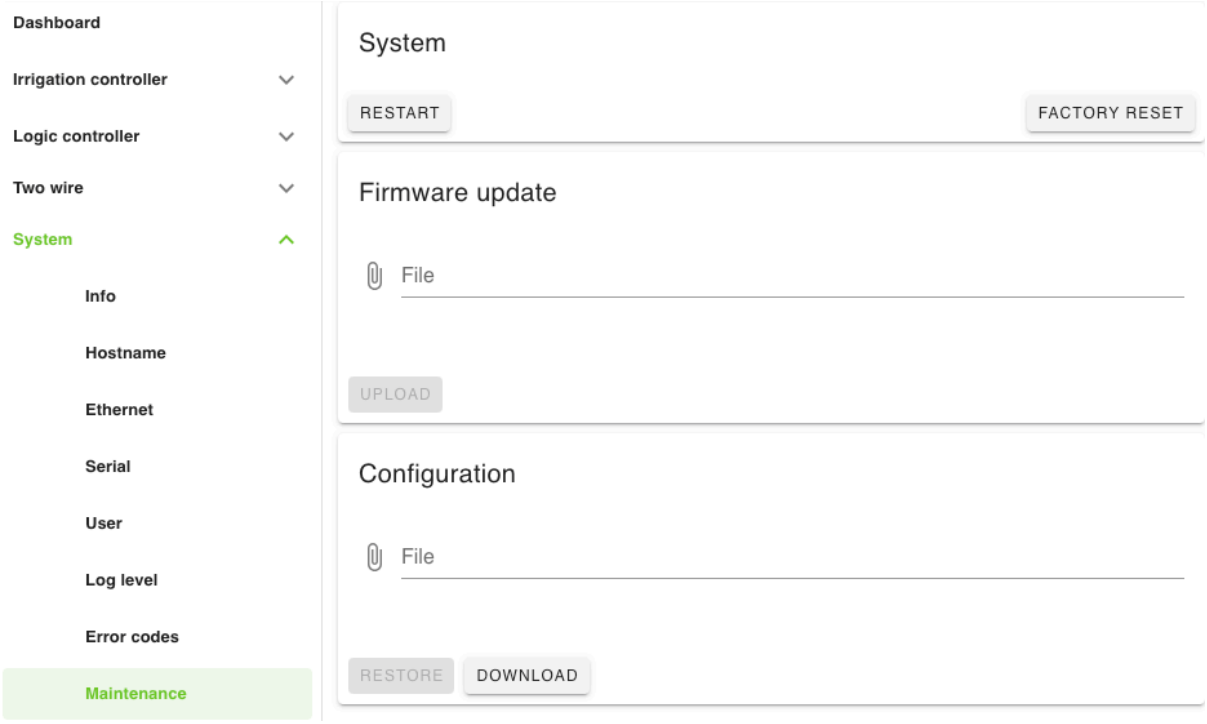


This setup ensures the sensor decoder sends interrupt when the reading changes.

Adding the logic to an existing setup:

The two files “” and “” contains the setup for fertigation and filter flush described above for differential and 2 input respectively.

Make a backup (download) of the configuration:



Open the downloaded json file with a plain text editor.
Localize the “le::engine” and “we::engine” texts. From and including the “le::engine” to and excluding the “we::engine” defines the logic engine. Substitute that section with the content of either of the two files.
Restore the configuration.