Digital Energy Meters

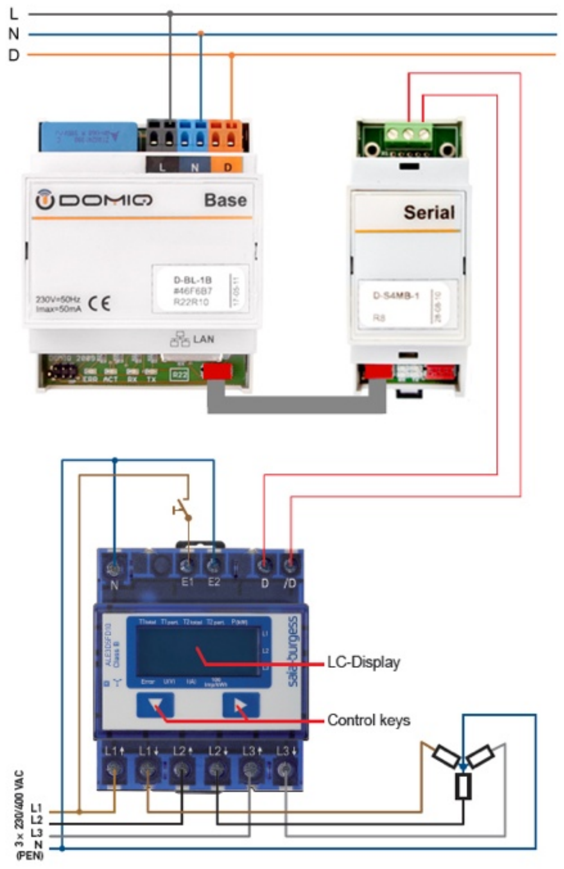
In this manual we will describe how to use **DOMIQ/Serial-4MB** module to integrate digital electricity meter **SAIA BURGESS ALE3D5F11** with the **DOMIQ/Base** and display measurement values in the **DOMIQ/Remote** application on the **iPhone** or on the **DOMIQ/ Display** touch panel.

In further part of this document you will find out how to:

- Read and display voltage values
- Read and display effective active power
- Read and display total energy counter
- Create resettable energy counter (hourly, daily, weekly, monthly measurement of electricity consumption)
- Signal power failure

DOMIQ devices comunicate with the meter using two-wire bus with MODBUS protocol. Up to 32 devices can be connected to single **DOMIQ/Serial-4MB**. Each device must have an individual addres in the range from 1 to 127. All devices also mus have configured the same transmission parameters (number of stop bits, parity type, transmission speed). Most of the available Modbus devices requires external power supply (typically 24V DC).

1. Wiring diagram



2. DOMIQ/Base module configuration

Each MODBUS device is seen as a set of 16-bit registers. Functions of the specific registers are not pre-defined, so always the device documentation is required. **DOMIQ/Base** and **DOMIQ/Serial-4MB** is a master MODBUS system. Each line in the configuration defines the device address, register number and the data format contained therein. Registers are queried periodically. Collected data is stored in the Base module and starts with the MODBUS prefix.

2.1. Creating the REMOTE menu

To display measured values on the iPhone you have to define the menu. To do this follow these steps:

1. Click on the **Remote** tab
2. Add a new **Page** and give it a name
3. Add a new **Section** e.g *Measurement values*
4. Add another **Section** e.g *Status*

2.2. Voltages

In first step you need to find (in meter documentation) the registers numbers where the values of voltage are stored. Next you can proceed to define registers reading.

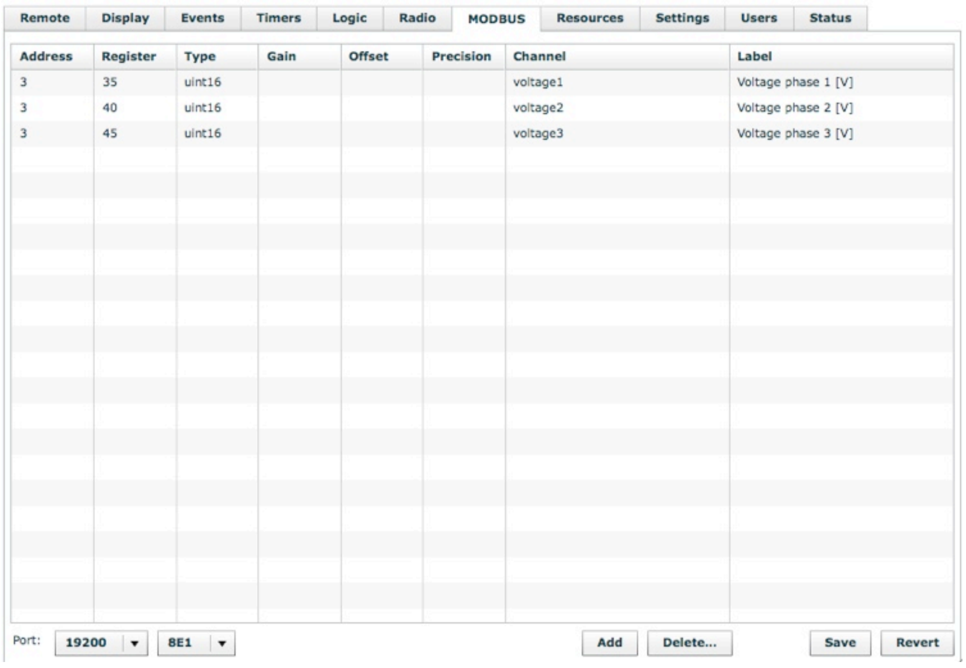
2.2.1. Registers Definitions

Please follow these steps:

1. After logging in to the **configurator**, select the **MODBUS** tab.
2. Click on the **Add** button to add a new register read.
3. In **Type** select register type. For 16-bit registers select **uint16**, for 32-bit registers select **uint32**. In this one choose **uint16** type.
4. In **Address** enter (in decimal) the device addres (it can be found in meter menu). In this example the meter address is **3**.
5. In **Register** type register address where the value of voltage on the first phase is stored. In this case it is register **36**. **Notice!** Some manufacturers gives the addresses starting with 1, and the part starting from 0. On the **MODBUS** tab addresses are counted from 0. In the SAIA meter adresses starts from 1, so in the register read enter **35** instead of 36.
6. In **Name** type a unique register name e.g *voltage1*. This name will be used to define the controls in the REMOTE application.
7. In **Label** add a short description of what the register readout concerns. e.g *voltage phase 1*
8. In the left bottom corner select transmission speed and parity control type. In this case :**19200** baud/s and parity control **8E1**.

For the remaining two phases repeat steps from 1. to 7. Remember to change the registers addresses to `40` and `45` at step **4.** and change the registers names to: `voltage2` and `voltage3.`

In result you will get:



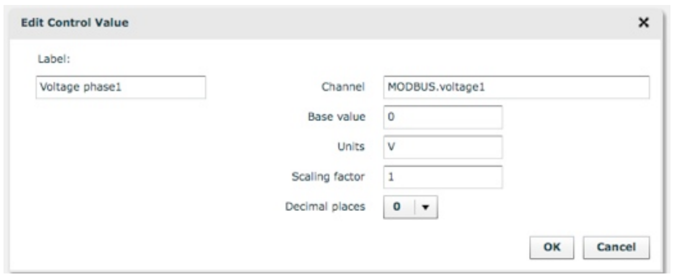
Address	Register	Type	Gain	Offset	Precision	Channel	Label
3	35	uint16				voltage1	Voltage phase 1 [V]
3	40	uint16				voltage2	Voltage phase 2 [V]
3	45	uint16				voltage3	Voltage phase 3 [V]

Port: 19200 8E1

2.2.2. Controls Definition

To preview the voltage values on the iPhone you have to define controls. To do this please follow these steps:

1. Click on the **Remote** tab
2. Find the menu structure created in section 2.1.
3. Add **Value** channel to the *Measurement values* section
4. Double-click on the new channel and make its features:
5. In **Channel** type `MODBUS.register_name`, where `register_name` is the content of the **Name** field in the **MODBUS** tab. In this example `MODBUS.voltage1`.

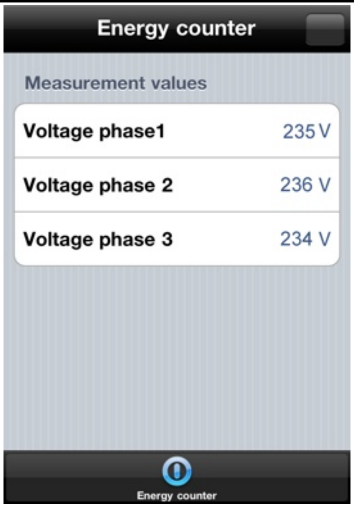

Edit Control Value

Label: Voltage phase1

Channel: MODBUS.voltage1
Base value: 0
Units: V
Scaling factor: 1
Decimal places: 0

For the remaining two phases repeat steps 3. to 5.

After saving the configurator settings you will get the following result on the iPhone:

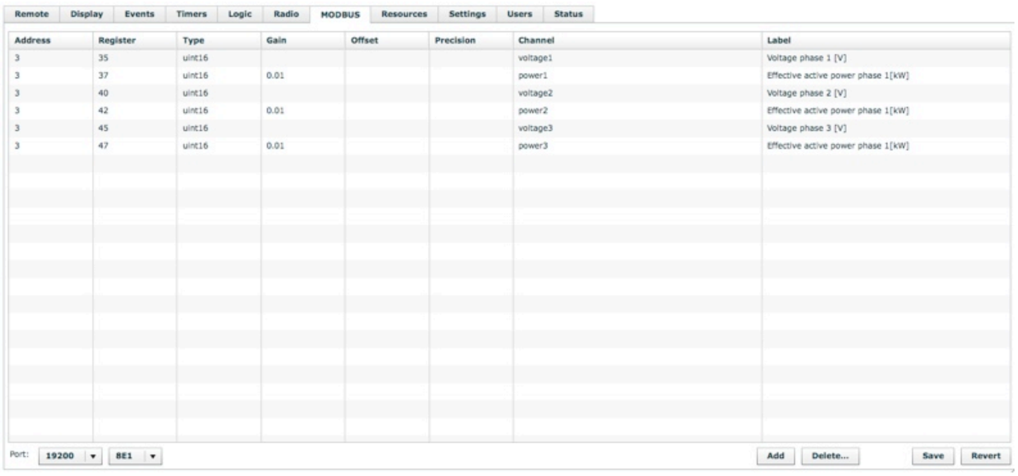


2.3. Effective Power

Read and display the effective active power is similar to the procedure presented in section 2.2. The only differences are:

2.3.1. Registers Definition

1. Enter the registers addresses where values of the effective active power are stored (in this case `35`, `42`, `47`).
2. Add **Gain** equal to `0.01` (in order to display floating point number)

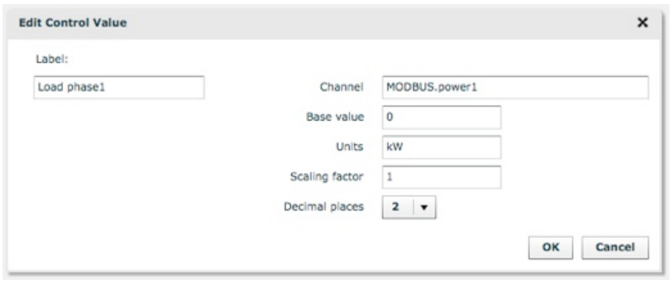


Address	Register	Type	Gain	Offset	Precision	Channel	Label
3	35	uint16				voltage1	Voltage phase 1 [V]
3	37	uint16	0.01			power1	Effective active power phase 1[kW]
3	40	uint16				voltage2	Voltage phase 2 [V]
3	42	uint16	0.01			power2	Effective active power phase 1[kW]
3	45	uint16				voltage3	Voltage phase 3 [V]
3	47	uint16	0.01			power3	Effective active power phase 1[kW]

Port: 19200 8E1

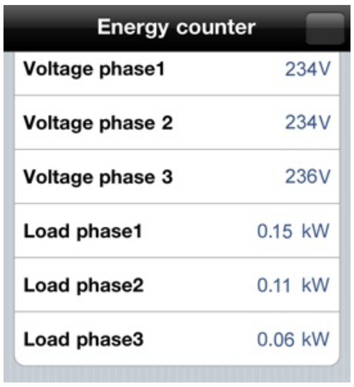
2.3.2. Controls Definition

1. Modify the **Channel** field in the **Value** channel. For example: if you used *power1* as a name of the register read, in the **Channel** field type `MODBUS.power1`.



Label:	Load phase1
Channel	MODBUS.power1
Base value	0
Units	kW
Scaling factor	1
Decimal places	2

The result on the iPhone:



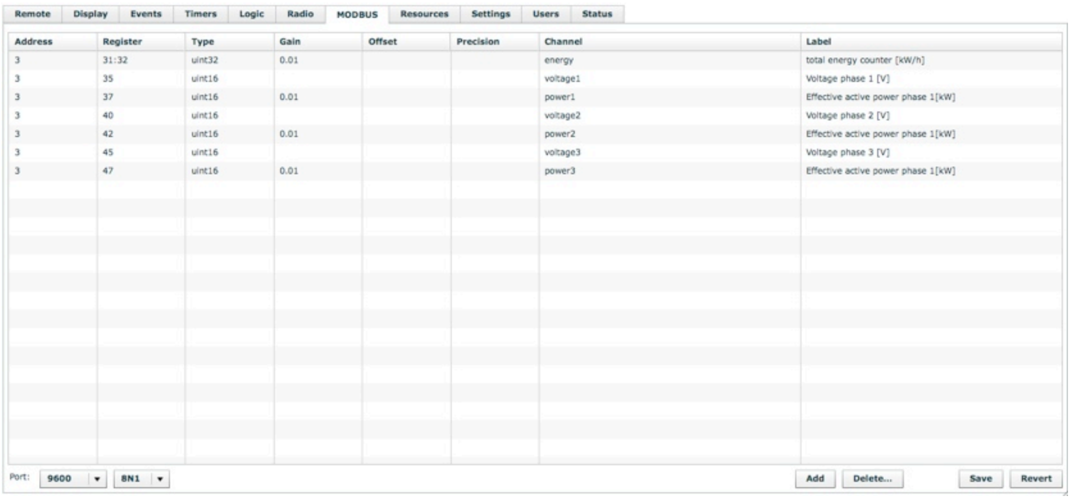
Energy counter	
Voltage phase1	234V
Voltage phase 2	234V
Voltage phase 3	236V
Load phase1	0.15 kW
Load phase2	0.11 kW
Load phase3	0.06 kW

2.4. The total energy counter

Read and display the total energy counter is similar to the procedure presented in section 2.2. The only differences are:

2.4.1. Registers Definition

1. Change register type to **uint32**
2. Add **Gain** equal to `0.01` (in order to display floating point number)
3. Enter the registers addresses. In this case it's a pair of registers: `31` and `32`
4. Add a unique name, for example *energy*

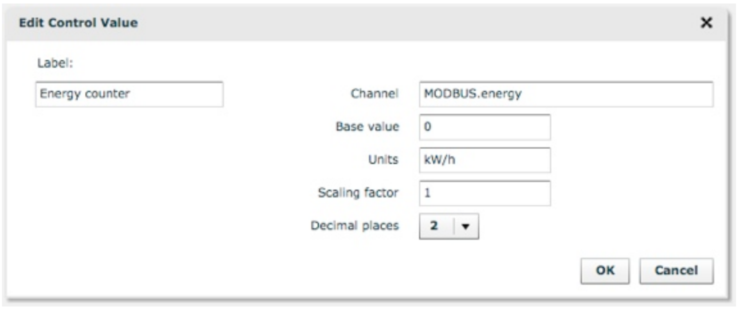


Address	Register	Type	Gain	Offset	Precision	Channel	Label
3	31:32	uint32	0.01			energy	total energy counter [kW/h]
3	35	uint16				voltage1	Voltage phase 1 [V]
3	37	uint16	0.01			power1	Effective active power phase 1[kW]
3	40	uint16				voltage2	Voltage phase 2 [V]
3	42	uint16	0.01			power2	Effective active power phase 1[kW]
3	45	uint16				voltage3	Voltage phase 3 [V]
3	47	uint16	0.01			power3	Effective active power phase 1[kW]

Port: 9600 8N1

2.4.2. Controls Definition

1. Modify the **Channel** field in the **Value** channel. For example: if you used *energy* as a name of the register read, in **Channel** type `MODBUS.energy.`



Label: Energy counter

Channel: MODBUS.energy

Base value: 0

Units: kW/h

Scaling factor: 1

Decimal places: 2

2.5. Resettable Energy Counter

The resettable energy counter allows for convenient measure of electricity consumption in any time period (hour, day, week, month etc.).

The simplest implementation of such meter requires to save the current value of the energy counter when user clicks on the **Reset** button. Next the saved value is subtracted from the current value and and the result of this subtraction is presented as a **resettable counter**.

In **DOMIQ/Base MEM** channel is used for permament variables storing. The value placed in this channel is stored until the next change. To store the current value of the meter is the best to use **VAR** channel - volatile variable.

To create the resettable energy counter follow procedure presented below.

2.5.1. Logic Script

1. In DOMIQ Base configurator click on the **Logic** tab
2. Paste there the source code shown below:

```
-- variables declaration --
counter = use 'MODBUS.energy'
save = use 'MEM.energy'
delta = use 'VAR.energy'

function clear()
  save:set(counter.value)
  delta:set(0)
end

function counter:onchange(new)
  delta:set(new - save.value)
end
```

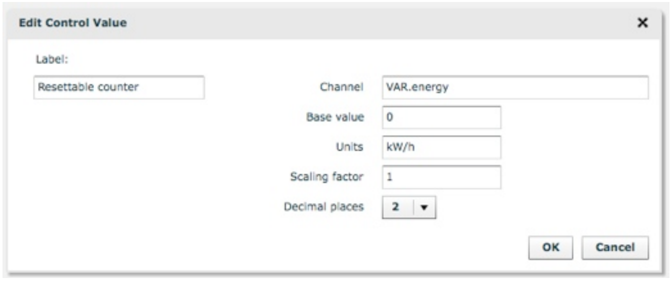
Function `clear` will be called after clicking on the **Reset** button. It is responsible for saving the current state of unresettable energy counter (variabl `save`) and reseting the value of the resettable energy counter (variable `delta`).

Function `counter` is called after every change of the total energy counter. This function calculates and saves (to the variable delta) difference between current value of the total energy counter and value of the total energy counter saved after clicking on the reset button.

2.5.2. Controls Definitions

1. Add **Value** channel to the *Measurement values* section in the **Remote** tab.
2. Double-click on the new channel and make its features.

IMPORTANT! In the **Channel** field enter the name of the variable that was defined in the **Logic** tab (in this case `VAR.energy`)

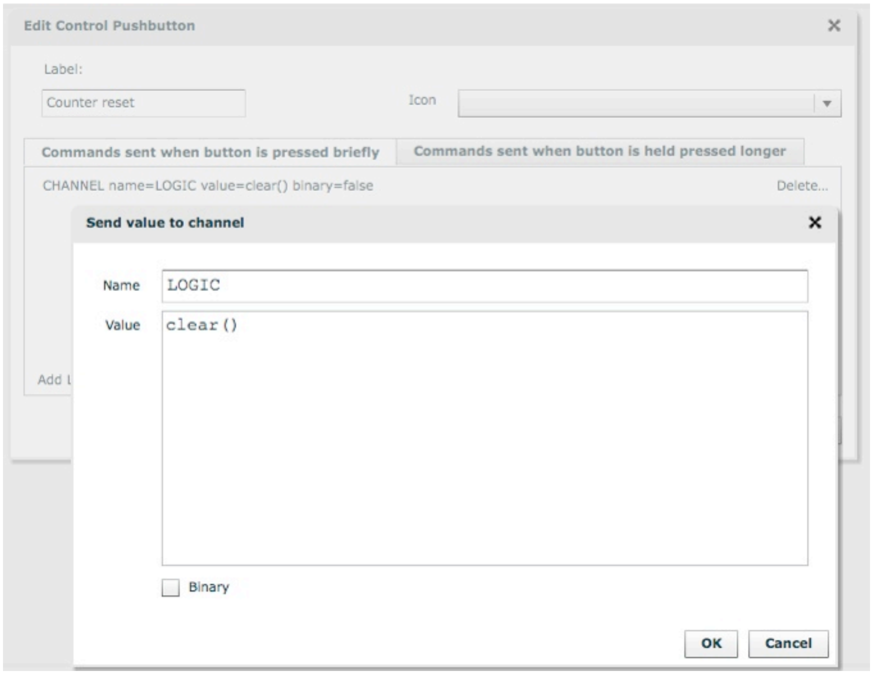


Edit Control Value

Label: Resettable counter

Channel: VAR.energy
Base value: 0
Units: kW/h
Scaling factor: 1
Decimal places: 2

3. Add **Pushbutton**
4. Double-click on it and make its features:
5. Give it a name e.g. *Counter reset*
6. in **hit** (for older software versions **Comands sent when button is pressed briefly**) tab click on the **Add channel** button and fill its features as shown below:



Edit Control Pushbutton

Label: Counter reset

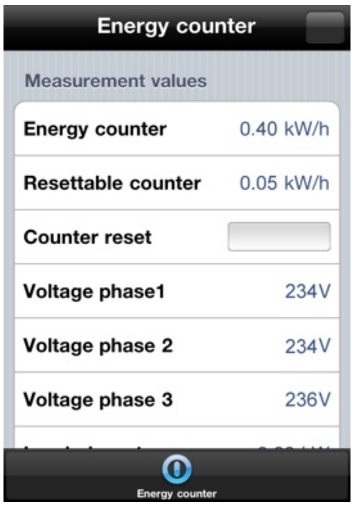
Commands sent when button is pressed briefly | Commands sent when button is held pressed longer

CHANNEL name=LOGIC value=clear() binary=false

Send value to channel

Name: LOGIC
Value: clear()
Binary: ☐

The result on the iPhone:



Energy counter

Measurement values

Energy counter	0.40 kW/h
Resettable counter	0.05 kW/h
Counter reset	
Voltage phase1	234V
Voltage phase 2	234V
Voltage phase 3	236V

2.6. Power State Notifications

When the power failure occurs you will receive a notification on the screen of the iPhone. It will also be visible in the Remote application.

Power failure is indicated when the measuered voltage falls below 180V. The additional VAR variable is created to signal the correct voltage in all phases.

2.6.1. Logic Script

Paste the source code presented below to the **Logic** tab:

```
-- variables declaration --
m_v1 = use 'MODBUS.voltage1'
m_v2 = use 'MODBUS.voltage2'
m_v3 = use 'MODBUS.voltage3'
v_v1 = use 'VAR.v1'
v_v2 = use 'VAR.v2'
v_v3 = use 'VAR.v3'
v_va = use 'VAR.va'

-- this function checks voltage on each phase --
function monitor()
  local v1,v2,v3 = 0,0,0
  if m_v1.value > 180 then v1 = 1 end
  if m_v2.value > 180 then v2 = 1 end
  if m_v3.value > 180 then v3 = 1 end
  v_v1:set(v1)
  v_v2:set(v2)
  v_v3:set(v3)
  if v1 + v2 + v3 == 3 then
    v_va:set(1)
  else
    v_va:set(0)
  end
end

m_v1.onchange = monitor
m_v2.onchange = monitor
m_v3.onchange = monitor
monitor()
```

Remember to use variable names that are identical to names used to read registers.

In variable declaration part seven variables was declared. The first three of them (m_v*) keep the voltage values on different phases. Remaing four (v_v*) were used for signaling power failure on the iPhone.

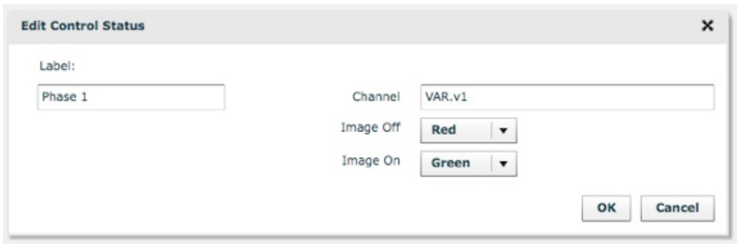
Function `Monitor` checks the voltage values and sets proper variables. It is called after any change of the voltage value.

2.6.2. Power State Controls

Power state controls will be represented as the lights that change color depending on the power state. It allows for a quick diagnose on which phase power failure occured.

To create power failure controls follow these steps:

1. Add **Status** channel to the *Status* section in the **Remote** tab.
2. Double-click on in and make its features as shown below:



Edit Control Status

Label: Phase 1

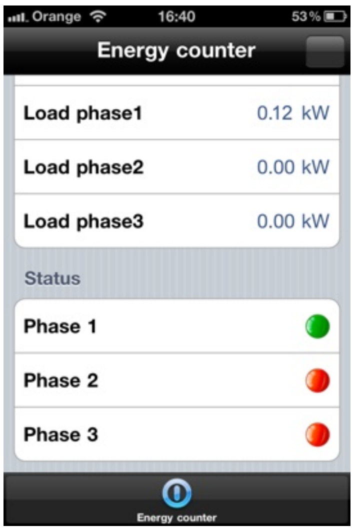
Channel: VAR.v1

Image Off: Red

Image On: Green

For the other two phases do the same, but remember to change the names of the variables to: `VAR.v2` and `VAR.v3`.

The result on the iPhone:



Energy counter

Load phase1	0.12 kW
Load phase2	0.00 kW
Load phase3	0.00 kW

Status

Phase 1	🟢
Phase 2	🔴
Phase 3	🔴

2.7. Power Fail and Restore Notifications

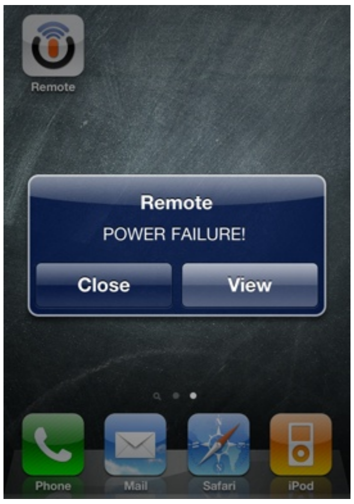
Notifications are displayed on all paired iPhones with **DOMIQ/Remote**. This section will describe how to use notifications to notify user about power failure and power restore.

2.7.1. Power Failure Notification

When power failure occurs you will receive a notification on the screen of the iPhone. To define such notification follow this procedue:

- Add an new event (click on the **Add** button) in the **Events** tab

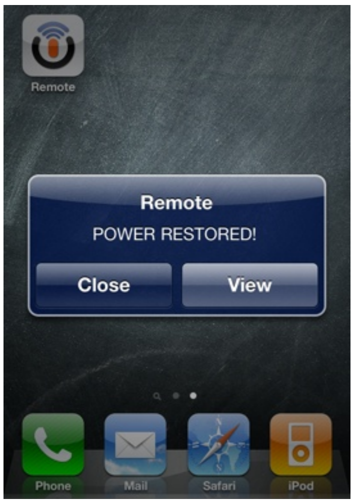
- In **Channel** type `E.VAR.va`
- Set match to **0**
- In **Command** click on **Add channel** and type in **Channel** `C.REMOTE.notify`.
- In **Value** type content of the notification that will be displayed on the iPhone screen. For example *POWER FAILURE!*



2.7.2. Power Restoration Notification

When power is restored you can also receive a notification on the screen of the iPhone. To define such notification follow this procedue:

- Add an new event (click on the **Add** button) in the **Events** tab
- In **Channel** type `E.VAR.va`
- Set match to **1**
- In **Command** click on **Add channel** and type in **Channel** `C.REMOTE.notify`.
- In **Value** type content of the notification that will be displayed on the iPhone screen. For example *POWER RESTORED!*



2.8. Visualization for the DOMIQ/Display

In this section we will show how to implement the functionality in the **DOMIQ/Display.**

In order not to increase the volume of this document we will only describe how to:

- Display voltages values
- Display power status
- Display power failure and power restoration notification

2.8.1. Visualisation Screen

1. In the **DOMIQ/Base** configurator select the **Display** tab.
2. From **New** list add a new **Screen** to **Tree** (drag and drop) and give it a unique name (e.g. *Energy meter).*

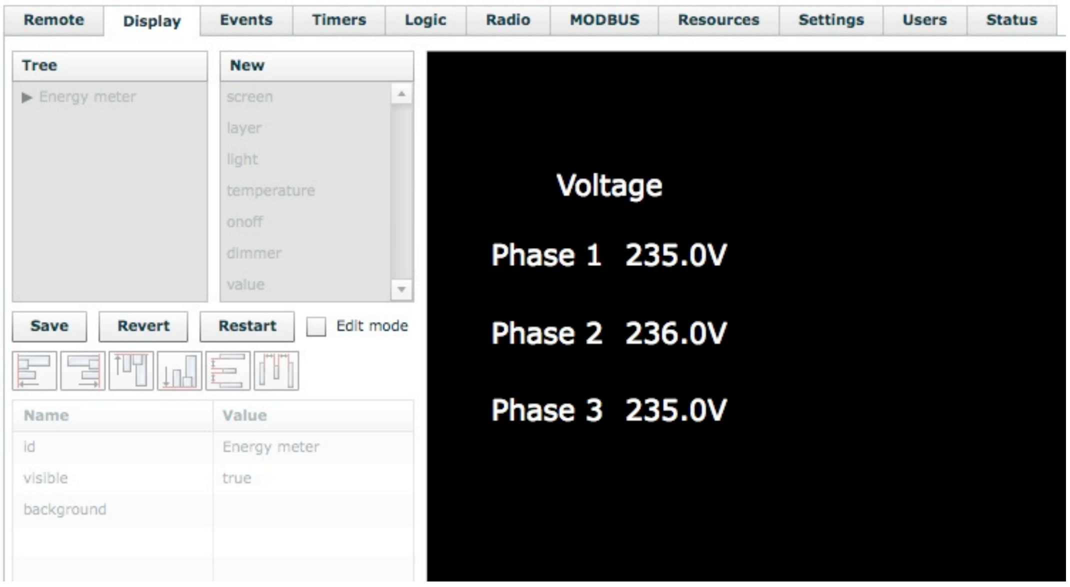
2.8.2. Voltage Values

To display the values of voltage follow the procedure presented below:

1. Add **Value** to the visualisation screen (drag it from **New** and drop it on the visualisation screen)
2. Click on it and type in **Channel**: `MODBUS.voltage1`
3. Add **Text** element to the visualisation screen
4. Click on it and type short description in the **text** field. For example *Phase 1*.
5. Add another **Text** element, click on it and type in the **text**: *Voltage*.

For the remaining two phases repeat steps from 2 to 4. Remember to change channel names to: `MODBUS.voltage2` and `MODBUS.voltage3` and controls descriptions to *Phase 2* and *Phase 3*.

The result is presented in the picture below:



Name	Value
id	Energy meter
visible	true
background	

2.8.3. Power Status

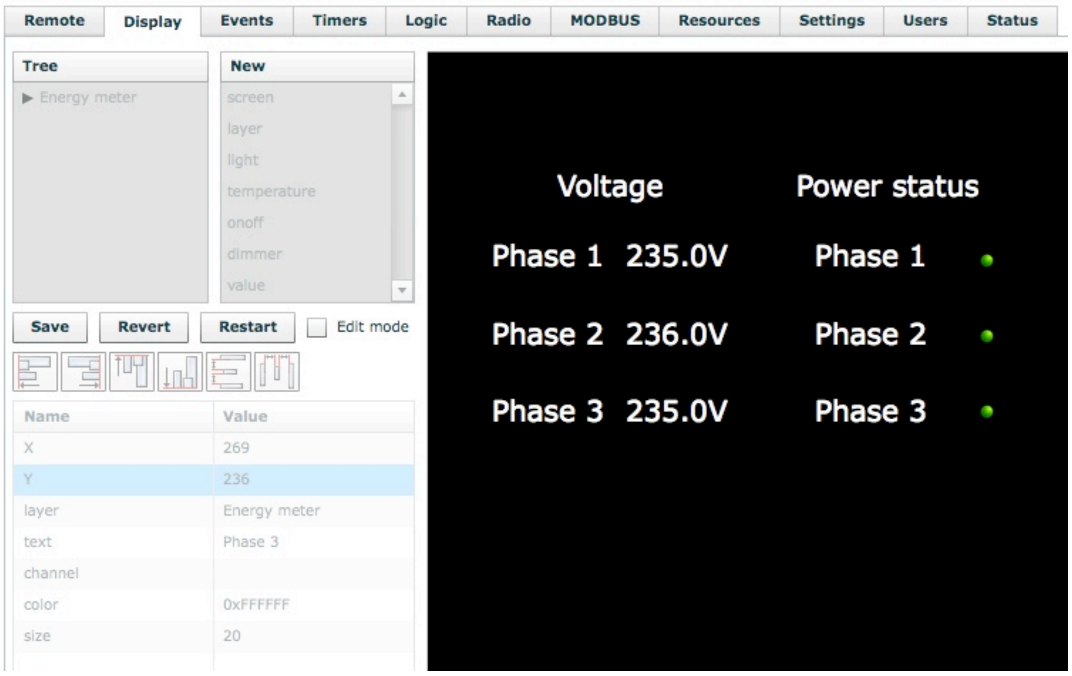
To display the power status please follow these steps:

1. Add **Light** to the visualisation screen
2. Click on it and type in the **channel**: `VAR.v1`

3. Choose **green** from **Theme** list
4. Add **Text** to the visualisation screen
5. Click on it and type short description in the **text** field. For example *Phase 1.*
6. Add another **Text** element, click on it and type in the **text**: *Power status*.

For the remaining two phases repeat steps from 2 to 4. Remember to change channel names to: `VAR.v2` and `VAR.v3` and controls descriptions to *Phase 2* and *Phase 3*.

The result is presented in the picture below:

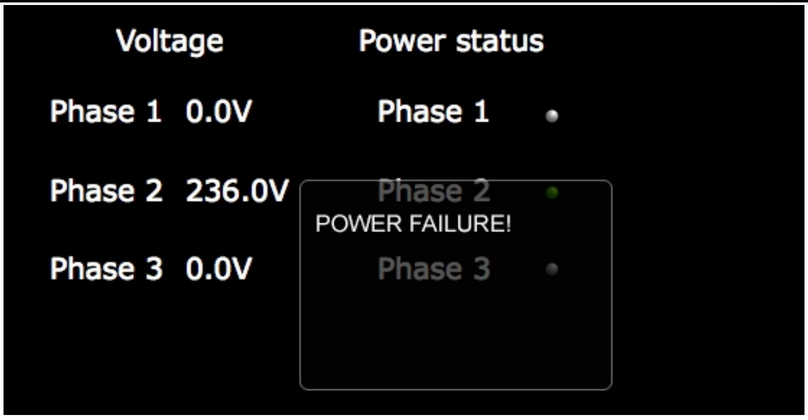


Name	Value
X	269
Y	236
layer	Energy meter
text	Phase 3
channel	
color	0xFFFFFF
size	20

2.8.4. Power State Messages

To receive a power failure notification on the **DOMIQ/Display** screen follow the procedure presented below:

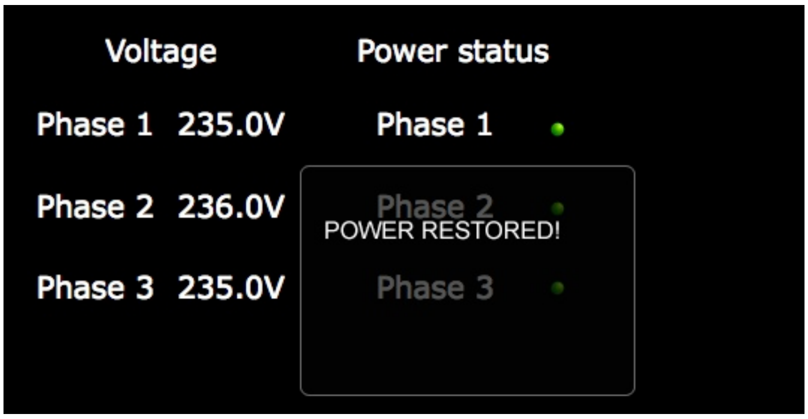
1. Click on the **Events** tab
2. Add a new channel (In **Command** click on the **Add Channel** button) to the earlier defined event (channel `E.VAR.va`, match `0`)
3. In **Name** type `C.DISPLAY.message`
4. In **Value** type content of the notification that will be displayed on the **DOMIQ/Display** touch screen. For example *POWER FAILURE!*



POWER FAILURE!

To receive a power restoration notification on the **DOMIQ/Display** screen follow the procedure presented below:

1. Click on the **Events** tab
2. Add a new channel (In **Command** click on the **Add Channel** button) to the earlier defined event (channel `E.VAR.va`, match `1`)
3. In **Name** type `C.DISPLAY.message`
4. In **Value** type content of the notification that will be displayed on the **DOMIQ/Display** touch screen. For example *POWER RESTORED!*



POWER RESTORED!