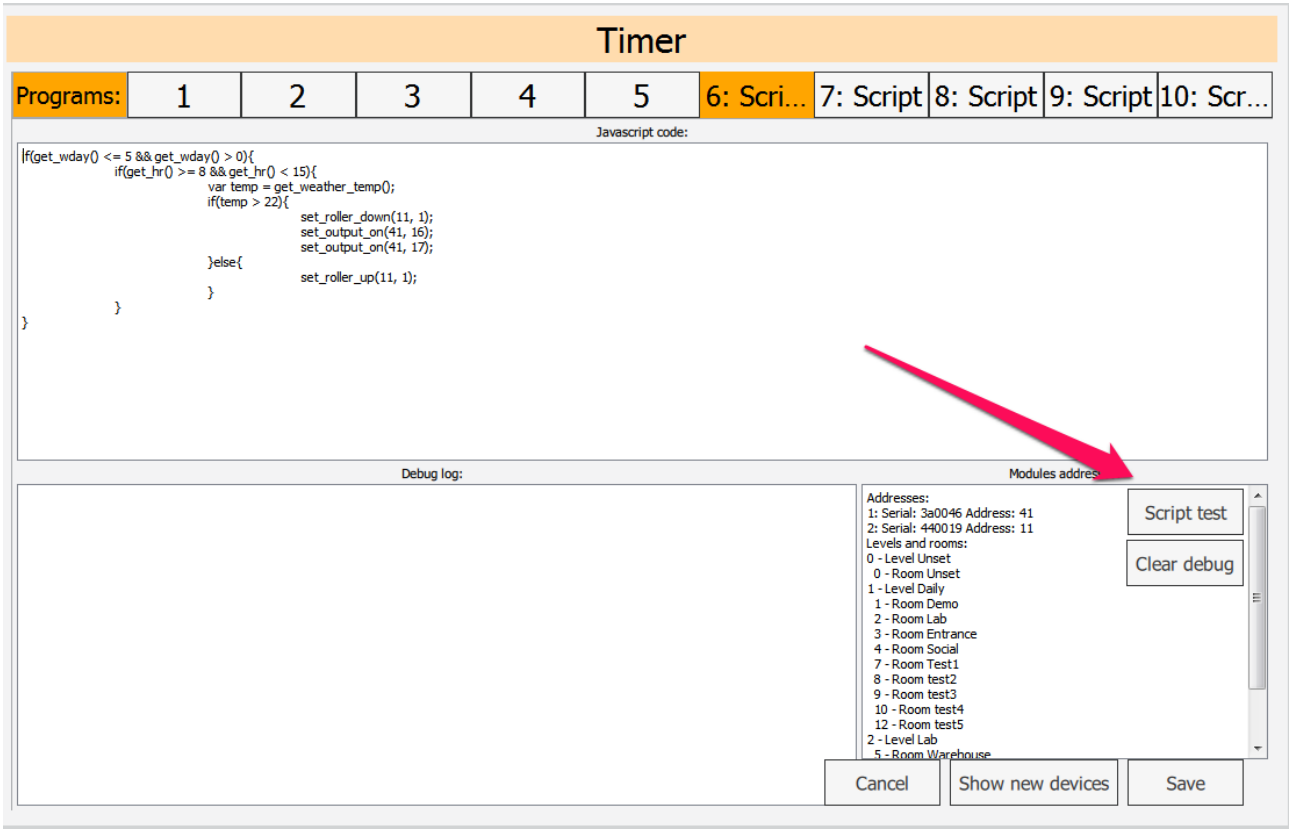
DEIMIC Scripting manual

Content:

Deimic API:

Always test your script before setting it(Picture 1.).



Picture 1. Test your script.

Language used in a scripts is Javascript. To declare variable use ‘var’ word(example ‘var a’).

Maxmimum length of a script is 4095.

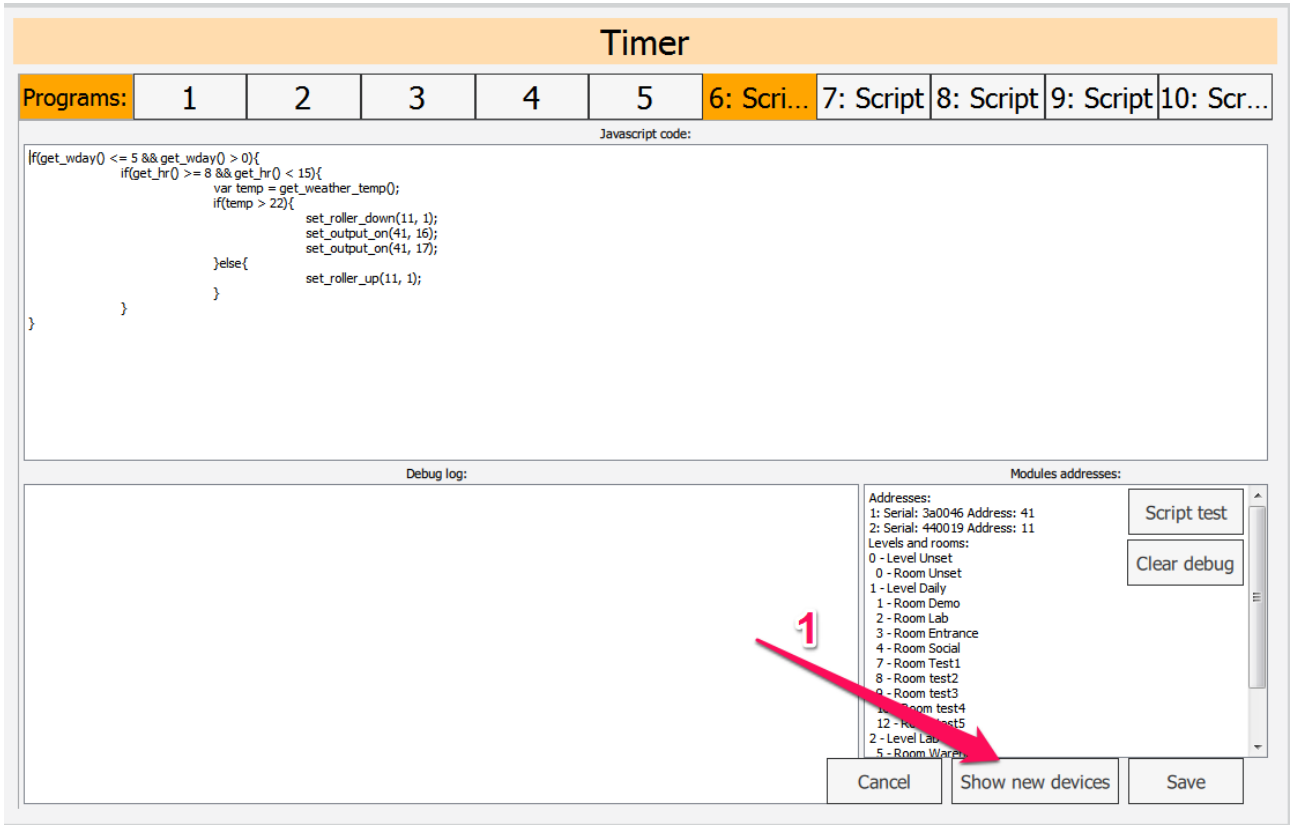
1. Javascript functions:

Math.abs(value) - absolute of given value
Math.round(value) - returnes nearest round of given value
Math.min(a,b) - returns minimum of two given values
Math.max(a,b) - returns maximum of two given values
Math.range(value,min,max) - returns value limited between two given values
Math.sign(value) - returns sign of given value (-1==negative,0=zero,1=positive)
Math.floor(value) - returns the largest integer less than or equal to a given number
Math.ceil(value) - returns the smallest integer greater than or equal to a given number
Math.PI() - returns PI value
Math.toDegrees(a) - returns degree value of a given angle in radians
Math.toRadians(a) - returns radians value of a given angle in degrees
Math.sin(a) - returns trig. sine of given angle in radians
Math.asin(a) - returns trig. arcsine of given angle in radians
Math.cos(a) - returns trig. cosine of given angle in radians
Math.acos(a) - returns trig. arccosine of given angle in radians
Math.tan(a) - returns trig. tangent of given angle in radians
Math.atan(a) - returns trig. arctangent of given angle in radians
Math.sinh(a) - returns trig. hyperbolic sine of given angle in radians
Math.asinh(a) - returns trig. hyperbolic arcsine of given angle in radians
Math.cosh(a) - returns trig. hyperbolic cosine of given angle in radians

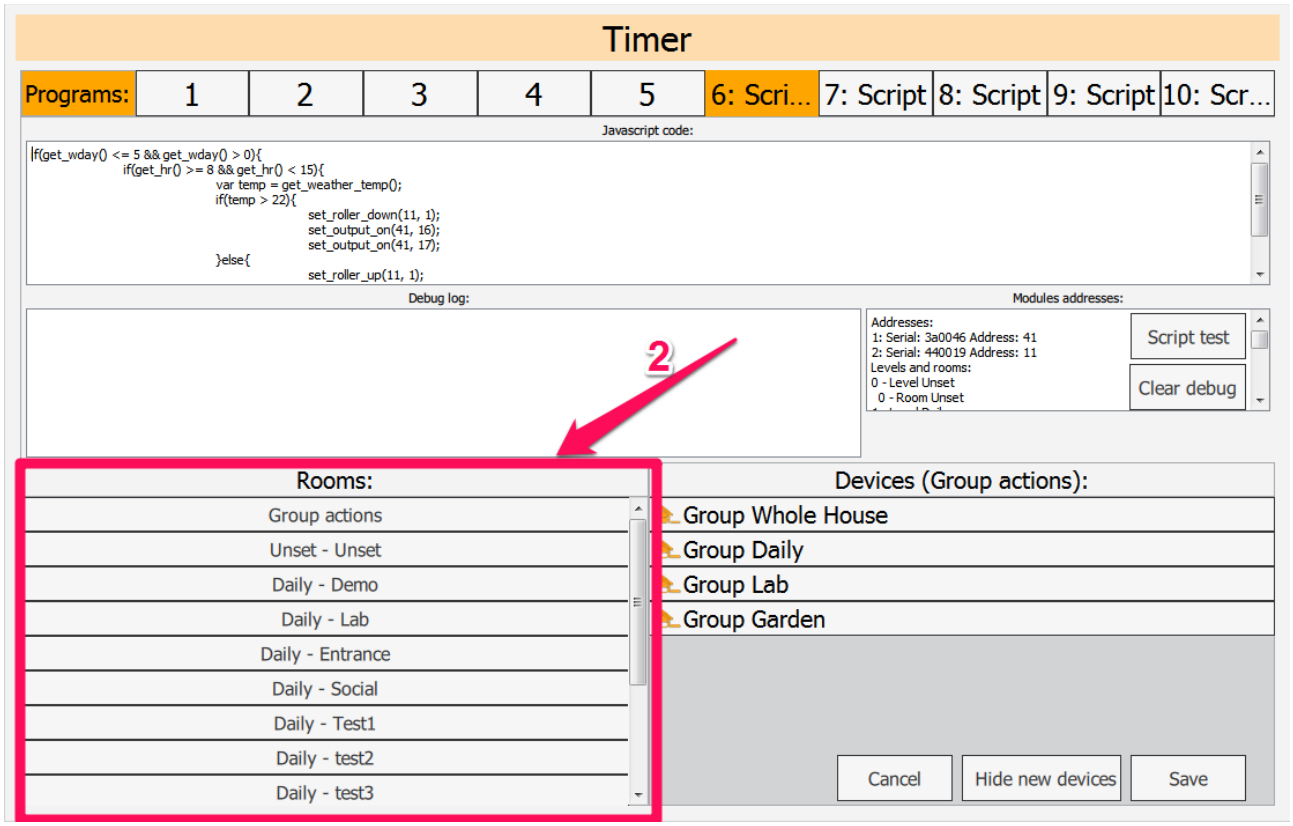
Math.acosh(a) - returns trig. hyperbolic arccosine of given angle in radians
Math.tanh(a) - returns trig. hyperbolic tangent of given angle in radians
Math.atan(a) - returns trig. hyperbolic arctangent of given angle in radians
Math.E() - returns E Neplero value
Math.log(a) - returns natural logaritm (base E) of given value
Math.log10(a) - returns logaritm(base 10) of given value
Math.exp(a) - returns e raised to the power of a given number
Math.pow(a,b) - returns the result of a number raised to a power (a)^(b)
Math.sqr(a) - returns square of given value
Math.sqrt(a) - returns square root of given value
Math.rand() - returns random value(maximum is INT_MAX)
Math.randInt(min, max) - returns random value between min and max
charToInt(ch) – changes char to int value
String.indexOf(search) - return index of search in a string, returns -1 if not found
String.substring(lo,hi) – creates substring lo is start, hi is stop
String.charAt(pos) – gets char at pos
String.charCodeAt(pos) – gets char code at pos
String.fromCharCode(char) – get string from char code
String.split(separator) – splits string in array of strings
Integer.parseInt(str) – parses int
Integer.valueOf(str) – tries to convert string into int
Array.contains(object) – checks if array contains object
Array.remove(object) – removes object from array
Array.join(separator) – creates string from array with seperator

2. Deimic functions:

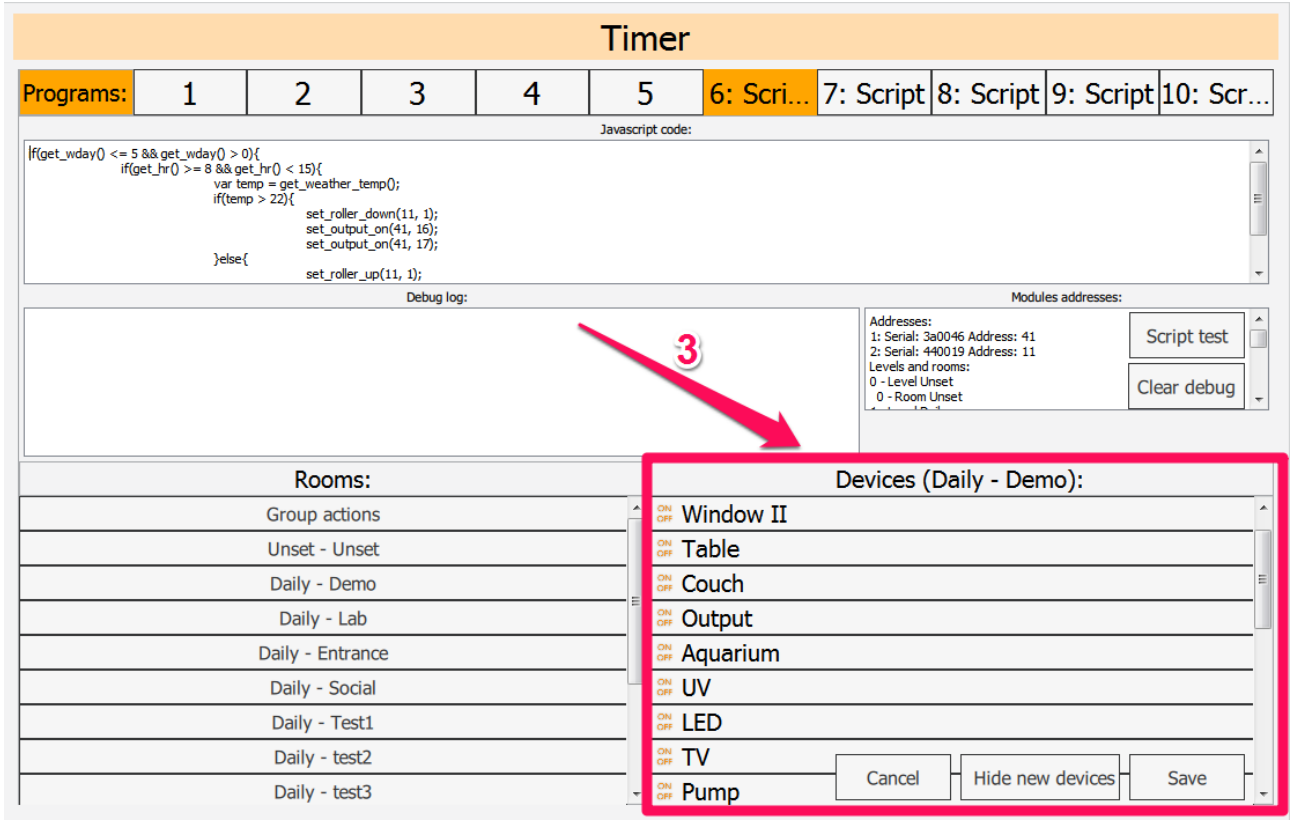
IMPORTANT:
To get output/roller/dimmer/recuperator/clicmaconvector address and number just click show new devices(1) at the bottom of the screen, select the room(2) and click on desired device(3) - it automatically writes address and number in javascript code



Picture 2. Show new devices.

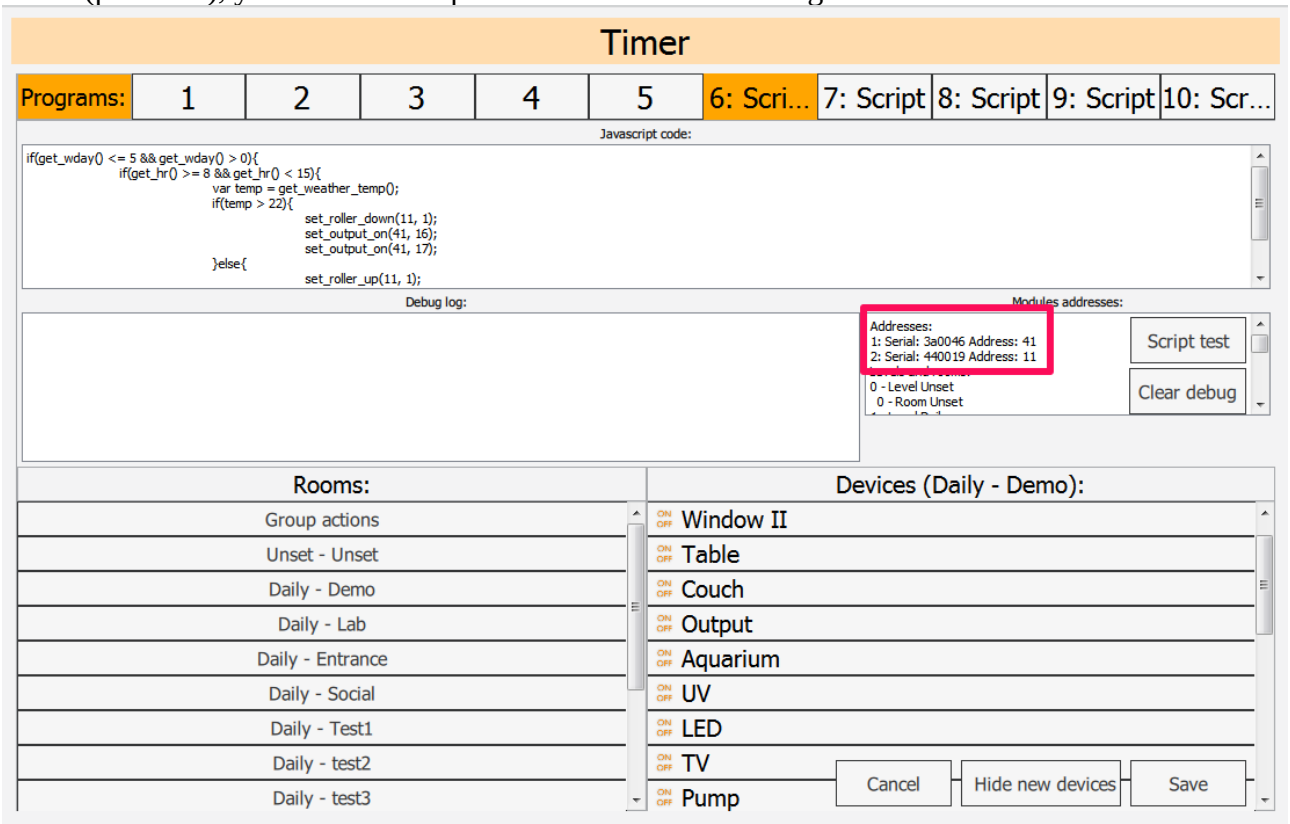


Picture 3. Select room.



Picture 4. Select device.

To get input address and number just check the address connected to serials on the right of the screen(picture 5), you can check input number in module configuration.



Picture 5. Address of a modules.

If you got DEIMIC ONE Green module, then inputs are group in two modules(addresses 524 and 525), first 32 inputs are in 524 module and rest 23 inputs are in 525 module. So input number 20 is 524,20 and input number 32 is 525,0, input number 40 is 525,8

sleep(seconds) - sleeps the thread for a number of seconds

---

msleep(msecons) - sleeps the thread for a number of miliseconds

---

get_serial() - gets system serial

2.1. GET FUNCTIONS:

---

get_input(address, number) - gets state of an input(0/1 short click, 2 long click) from a module address and input number - all known addresses are shown on the right side of the screen
returns -1 if device is not available

---

get_output(address, number) - gets state of an output from a module address and output number
returns -1 if device is not available

---

get_dimmer(address, number) - gets state of a dimmer(value between 0% and 100%) from a module address and dimmer number
returns -1 if device is not available

---

get_roller(address, numer) - get state of a roller from a module address and roller number
return -1 if device is not available
return 0 - state unknown/stop
return 1 - state stop
return 2 - state run up
return 3 - state run down
return 4 - state up
return 5 - state down
return 6 - state wait(it is the state when roller is running up/down and we click to run the other direction and it stops for a second)

Available since 05.02.2018
get_recuperator(address, numer) - get state of a roller from a module address and roller number
return -1 if device is not available
return 0/1/2/3/4 – gear num

If recuperator is connected to 0-10V output use get_dimmer function

---
Available since 29.04.2019
get_water(address, number) – get state of a watering
reteurn -1 if device is not available

Available since 29.04.2019
get_heat(address, number) – get state of a heat section(1 – heating is on, 0 – heating is off)
reteurn -1 if device is not available

2.2. OUTPUTS:

---

set_output_on(address, number) - sets on selected output, returns void

---

set_output_off(address, number) - sets off selected output, returns void

---

set_output_toggle(address, number) - toggles the output(changes the state between on and off), returns void

---

set_output_set(address, number, time1, time2, future_use) - blinks the output for a count number, time1 is off time, time2 is on time, if you want to turn on the light for a specific time set time2 and count to 0. Time unit is 0.1 seconds, so 10 is 1 seconds.
Last parameter is not used for now, you should set it to 0.
Example: set_output_set(1, 5, 0, 600, 0) – turns the light on for 60 seconds.

---

set_group(group, level, room, subgroup, value) - sets value to a group of devices(outputs and dimmers) - group of devices we want to change coded in bits, levels coded in bits, rooms coded in bits, subgroup coded in bits, value(from 0 - turn off to 100 - dimmer maximum value)

Available groups:
1 - Lights
2 - Sockets/Radio/etc.
3 - Gates
4 - Outside lights
5 - LEDs
6 - Technical

So to send to a specific group we need to set correct bits. We want to send to lights group so we set 1st bit (1 << 1) which is 0b00000010
We can also combine groups together example lights, outside lights and leds (1 << 1) | (1 << 4) | (1 << 5) which is 0b00110010

Subgroup we need to set to 0xFF

We can see level codes on the right of the screen, same for rooms.
So to set for a specific level just move the bit with (1 << LEVEL_ID) and room id to 0xFFFFFFFE(0 bit is for unused level and room, we do not want to set unused devices)

For setting only one room we need to find level and room ids and set them correctly level = (1 << LEVEL_ID) and room = (1 << ROOM_ID)

2.3. DIMMERS:

---

set_dimmer_set(address, number, value, time) - sets dimmer to a specific value(0-100%) in a specified time(1 time means 0.01s, so to set it to 1s time should be set to 100), it is time to reach the value, it changes the value smoothly.

---

set_dimmer_settime(address, number, value, time, time2) - sets dimmer to a specific value(0-100%) in a specified time2(1 time means 0.01s, so to set it to 1s time should be set to 100), it is time to reach the value, it changes the value smoothly, after time2 dimmer changes value to 0%
Time2 is in seconds. This command may be used to set dimmer for time, for example use this command on motion sensors.

---

set_dimmer_toggle(address, number, value, time) - toggle dimmer between value(1 - 100%) and 0% in a specified time(1 time means 0.01s, so to set it to 1s time should be set to 100), it is time to reach the value, it changes the value smoothly,

---

set_dimmer_wavergb(address, number, time) - starts/stop rgb wave, time 1 is equal to 760ms

---

set_dimmer_wave(address, number, time) - starts wave for 1 dimmer, time 1 is equal to 230ms

---

set_dimmer_set3x(address, value_red, time_red, value_green, time_green, value_blue, time_blue) - sets the value of a RGB, value and times are set in the same way as in set_dimmer_set command

---

2.4. ROLLERS:

---

set_roller_upstop(address, number) - run roller up or stops it if running

---

set_roller_downstop(address, number) - run roller down or stops it if running

---

set_roller_up(address, number) - run roller up

---

set_roller_down(address, number) - run roller down

---

set_roller_stop(address, number) - stops the roller

---

set_roller_step(address, number) - changes the state of a roller in order(run up, stop, run down, stop)

---

set_roller_groupup(group,level,room,subgroup) - sets group of rollers to run up, levels and rooms are the same as in set_group command, available groups and subgroups are shown below set_roller_groupstop command

---

set_roller_groupdown(group,level,room,subgroup) - sets group of rollers to run down, levels and rooms are the same as in set_group command, group and subgroups are shown below set_roller_groupstop command

---

set_roller_groupstop(group, level, room, subgroup) - set group of rollers to stop, levels and rooms are the same as in set_group command

Avaialable groups:

1 - Roller
2 - Shutter facade
3 - Unused
4 - Marquise
5 - Projector

Available subgroup(cardinal direction):

0 - N
1 - NE
2 - E

3 - SE
4 - S
5 - SW
6 - W
7 - NW

Set group and subgroup in the same way as in set_group command.
Example:
We want to chagne state of a rollers, suhtter facades and marquises and subgroup set to N, NE or NW. So group value would be 0b00010110
and subgroup 0b10000011

2.5. RECUPERATOR:

---

set_recuperatorset(address, number, value) - set recuperator gear to value

---

set_recuperatorstop(address, number) - stop recuperator

2.6. CLIMACONVECOTR:

---

set_climaon(address, number) - turns on climaconvector to previous mode

---

set_climaoff(address, number) - turns off climaconvector

2.7. HEATING:

---

set_heatprofile(address, number, value) - set heat profile, value can be in range:
- 0 - OFF
- 1 - ECO
- 2 - COMFORT
- 3 - MAX

---

get_temperature(address, number) - temperature of heating section
returns -100 if error

---

get_temperature_room(level, room) - temperature of specific level and room
available since version 22.11.2016r
returns -100 if error

2.8. GLOBAL AND DEBUG:

---

set_global(number, value) - sets global variable(number is the name of variable) to value, it can be used for example to block the script from running twice, at first line of script check global and after that set it to 1, so if next script starts it stops.

Remember this variable are set to 0 after every reboot of the system, so they are no good for saving some data

---

get_global(number) - gets global variable(number is the name of variable)

---

set_global_file(number, value) - sets global variable(number is the name of variable), the difference is that this variables are saved in a file, so after reboot values are restored

---

get_global_file(number) - gets global variable(number is the name of variable)

---

printstr(str) - prints string value to debug console, works only in testing mode

---

printnum(num) - prints int value to debug console, works only in testing mode

---

printfloat(float) - prints float value to debug console, works only in testing mode

---

2.9. NETWORK TCP AND UDP CONNECTIONS

---

send_tcp(address, port, data) - sends string data to a tcp server, address of a host, port of a host, data(string) to send, returns number of bytes written

---

receive_tcp(address, port, length) - connects to tcp, waits for data(length), address of a host, port of a host

---

send_receive_tcp_time(address,port,data,time) – connect to tcp, sends data, wait time in ms and returns all received data

---

send_receive_tcp_time(address, port, data, time) – connects to server using tcp, sends data, wait time and returns data received(time in 100ms)

---

send_receive_tcp_http_get(address, data, length) - connects to http server and reads length of data, address of a host, data to send to host, GET method arguments

---

send_tcp_http_get(address) - connects to http server, used when read is not needed

---

send_udp(address, port, data) - sends string data to udp server, address of a server, port of a server, data(string) to send

---

send_udp_time(address, port, data, time, get_port) - sends string data from port get_port to udp server, address of a server, port of a server, data(string) to send and waits time[100ms] for data, which is returned, this function was added on 14.03.2017

2.10. TIME

---

get_light() - gets value if it is day or night(1 - day, 0 – night), day is between sunrise and sunset, it is calculated using geocoordinates

---

get_time_t() - gets current time_t - seconds since 1 Jan 1970

---

get_min() - gets current number of minutes

---

get_hr() - gets current hour

---

get_wday() - gets weekday (0 - Sunday, 1 - Monday, 2 - Tuesday, 3 - Wednesday, 4 - Thursday, 5 - Friday, 6 - Saturday)

---

get_mday() - gets day of a month

---

get_yday() - gets day of a year

---

get_year() - gets current year number

---

get_time() - gets time a double form(example 1:30 PM is 13,5 - hour 13 and 0,5 as 30 minutes)

---

get_sunrise_time() - gets sunrise time in double form same as in get_time() command

---

get_sunset_time() - gets sunset time in double form same as in get_time() command

---

2.11. WEATHER

Check if return value is not -100(error state).

---

get_weather_temp() - gets current weather temperature in double in celcius

---

get_weather_temp_min() - gets todays weather minimum temperature

---

get_weather_temp_max() - gets todays weather maximum temperature

---

get_weather_wind_speed() - gets current wind speed in m/s

---

get_weather_wind_dir() - A numerical value representing the direction that the wind is coming from in degrees, with true north at 0° and progressing clockwise

---

get_weather_raining() - binary value, 1 - currently raining, 0 - currently no rain

---

get_weather_cloud() - A numerical value between 0 and 1 (inclusive) representing the percentage of sky occluded by clouds.

---

get_weather_pressure() - A numerical value representing the sea-level air pressure in hPa

---

get_weather_humidity() - A numerical value between 0 and 1 (inclusive) representing the relative humidity

---

2.12. MODBUS/RS485

Modbus commands are available since hardware version 6.0.

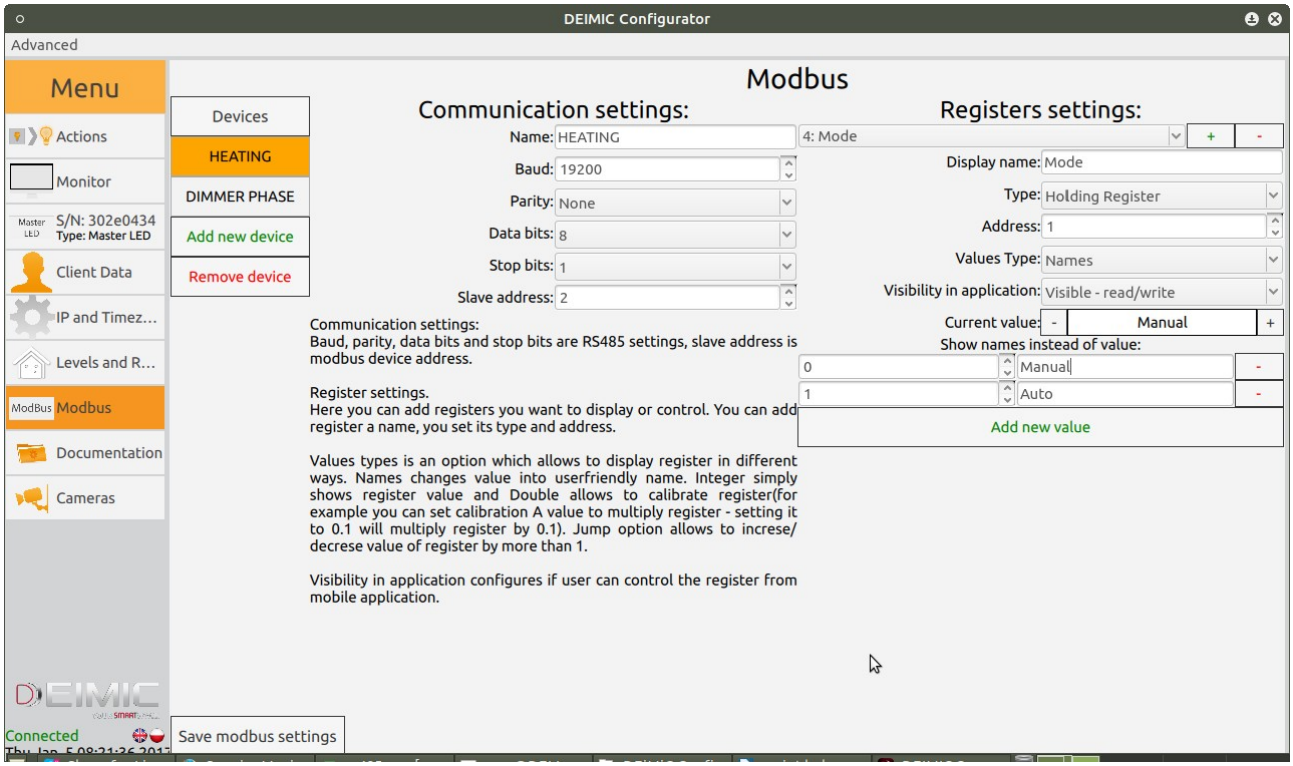
Register Type:
ModbusCoil = 0
ModbusDiscreteInputs = 1
ModbusInputRegister = 2
ModbusHoldingRegister = 3

---

get_modbus_register(device_number, register_address, register_type) – returns the value of the register if available and -1 if error. Device_number is a device number in system counter from 0, check picture 6 for more information. Register address is modbus register address. Available types are listed above. IMPORTANT – get_modbus_register returns array, so if the length of the register is 1 just use: get_modbus_register(device_number, register_address, register_type)[0]

---

set_modbus_register(device_number, register_address, register_type, value) – sends command to set modbus register, check get_modbus_register function for information about parameters, value parameter is a intiger value we want to set.



Picture 6. We can see 2 MODBUS devices available. Heating(device_numer – 0) and DIMMER_PHASE(device_number – 1).

2.13. COUNTERS

Counters commands are available from version 2017-02-23

---

get_counter_value(addr, input) – returns the value of the impulse counter from a module(addr) and input number(input).

2.14 SONOS

Ask your reseller if it is available in your version of DEIMIC module.

To control sonos use send_udp(address, port, data) command with specific address(”127.0.0.1”), port(5010) and data corresponding correct command. Data has specific order shown below:

- play command:
  send_udp(”127.0.0.1”, 5010, ”sonos;;play;;<zone name>”);
  where <zone name> is sonos zone name

- play/pause command:
  send_udp(”127.0.0.1”, 5010, ”sonos;;play_pause;;<zone name>”);
  where <zone name> is sonos zone name

- pause command:
  send_udp(”127.0.0.1”, 5010, ”sonos;;pause;;<zone name>”);
  where <zone name> is sonos zone name

- stop command:
  send_udp(”127.0.0.1”, 5010, ”sonos;;stop;;<zone name>”);
  where <zone name> is sonos zone name

- next track command:
  send_udp(”127.0.0.1”, 5010, ”sonos;;next;;<zone name>”);
  where <zone name> is sonos zone name

- previous track command:
  send_udp(”127.0.0.1”, 5010, ”sonos;;previous;;<zone name>”);
  where <zone name> is sonos zone name

- mute command:
  send_udp(”127.0.0.1”, 5010, ”sonos;;mute;;<zone name>”);
  where <zone name> is sonos zone name

- unmute command:
  send_udp(”127.0.0.1”, 5010, ”sonos;;unmute;;<zone name>”);
  where <zone name> is sonos zone name

- mute/unmute command:
  send_udp(”127.0.0.1”, 5010, ”sonos;;mute_change;;<zone name>”);
  where <zone name> is sonos zone name

- volume set command:
  send_udp(”127.0.0.1”, 5010, ”sonos;;volume;;<zone name>;;<volume level>”);
  where <zone name> is sonos zone name and <volume level> is volume level to set

- increment volume command:

  send_udp(”127.0.0.1”, 5010, ”sonos;;volume_inc;;<zone name>;;<volume increment>”);
  where <zone name> is sonos zone name and <volume increment> is how many percent to increment

- decrement volume command:

  send_udp(”127.0.0.1”, 5010, ”sonos;;volume_dec;;<zone name>;;<volume decrement>”);
  where <zone name> is sonos zone name and <volume decrement> is how many percent to increment

- set favorite list:

  send_udp(”127.0.0.1”, 5010, ”sonos;;favorite;;<zone name>;;<favorite name>”);
  where <zone name> is sonos zone name and <favorite name> is favorite name

- set playlist:

  send_udp(”127.0.0.1”, 5010, ”sonos;;playlist;;<zone name>;;<playlist name>”);
  where <zone name> is sonos zone name and <playlist name> is playlist name

- say text:

  send_udp(”127.0.0.1”, 5010, ”sonos;;say;;<zone name>;;<text to say>;;<language>;;<volume level>”);
  where <zone name> is sonos zone name, <text to say> is text, <language> is

language in which system should say the text and <volume level> is volume level set before saying text(0 for no change). For example send_udp(”127.0.0.1”, 5010, ”sonos;;say;;Test;;Welcome home;;en”;;0); to play ”Welcome home” text in english language in “Test” zone with no volume change

- say text and restore the track:

send_udp(”127.0.0.1”, 5010, ”sonos;;say_restore;;<zone name>;;<text to say>;;<language>;;<volume level>”);
where <zone name> is sonos zone name, <text to say> is text, <language> is language in which system should say the text and <volume level> is volume level set before saying text(0 for no change). For example send_udp(”127.0.0.1”, 5010, ”sonos;;say;;Test;;Welcome home;;en;;0”); to play ”Welcome home” text in english language in “Test” zone with no volume change

IMPORTANT: You can only restore to saved playlist/favorite

- create snapshot of sonos speaker:

send_udp(”127.0.0.1”, 5010, ”sonos;;snapshot;;<zone name>);
where <zone name> is sonos zone name, you can later restore the state using restore command, it is very useful to for example save state, change volume, play some “text to speech” and restore the state to the previous one

IMPORTANT: You can only create snapshot of saved playlist/favorite

- restore the state of sonos speaker from snapshot:

send_udp(”127.0.0.1”, 5010, ”sonos;;restore::<zone name>);
where <zone name> is sonos zone name to restore state

IMPORTANT: You can only restore to saved playlist/favorite

2.15 NOTIFICATION

- notification_send(<num>)

This commands sends specific notification. By clicking on notification in “add new device” tab you can add this command automatically.
Example: notification_send(2);

- notification_send_text(<num>, <title>, <text>)

This commands allows to send notification to devices or email and replace title and text. By allowing title and text swap it is possible to send some parameters with notifcation, for example system time or motion sensor name.
Example: notification_send_text(2, “Motion sensor”, “Move detected “ + get_hr() + “:” + get_min());

2.16 WATERING

- set_water_block(<time>)

Blocks watering to specific unix timestamp
Example set_water_block(get_time_t() + 24 * 60 * 60); - blocks watering for 24 hours

- set_water_on(<module address>, <water num>, <time>)

Starts watering, water num is between 0 and 9, time between 1-255, time is in minutes
Example set_water_on(1, 1, 3); - starts watering for 3 minutes

- set_water_off(<module address>, <water num>)

Turns off watering section if on, water num is between 0 and 9
Example set_water_off(1, 1);

3. Examples

It is good to write some print functions just to be sure if the script works. After the script is finished you should remove this nums so they will not be send next time.

3.1. Close the roller and turn on the lights if it is too hot outside in a specific days and time

```
//This script should be set in a timer
//(11, 1) - is the roller address
//(41, 16) - is light address

if(get_wday() <= 5 && get_wday() > 0){ //If it is weekday
        if(get_hr() >= 8 && get_hr() < 16){ //Between 8 AM and 4 PM
                var temp = get_weather_temp(); //Read current temperature outside
                if(temp > 22){ //If temperature is more than 22 celcius degrees
                   set_roller_down(11, 1); //Run down the roller
                   set_output_on(41, 16); //turn on the light
                }else{ //If temperature goes down then open the roller
                   set_roller_up(11, 1);
                }
        }
}
```

3.2. Send data using UDP

Command below sends data to IR tranmitter.

```
send_udp("192.168.50.103", 7777, "@IR_SEND@100;100;100;" + String.fromCharCode(13) + String.fromCharCode(10));
//String.fromCharCode(13) is carriage return(\r – code 13)
//String.fromCharCode(10) is new line(\n – code 10)
```

3.3. Send and read data using TCP – DENON Mute On/MuteOff

send_receive_tcp_time

//Sends data and waits for 100ms(last parameter) and reads data

```
var string = send_receive_tcp_time("192.168.50.201", 23, "MU?" + String.fromCharCode(13), 1);

msleep(100); //Need to wait to create new tcp connection
if(string.indexOf("MUOFF") >= 0) //Check if string contains "MUOFF"
	send_tcp("192.168.50.201", 23, "MUON" + String.fromCharCode(13)); //If contains than turn mute
on
else
	send_tcp("192.168.50.201", 23, "MUOFF" + String.fromCharCode(13)); //Else turn mute off
```

3.4. Write to modbus register and verify it

```
var time_maximum = 10;//Maximum waiting time in secs
var valueToSet = 0;
var deviceNum = 0;
var registerAddress = 1;
var registerType = 3;

if(get_modbus_register(deviceNum, registerAddress, registerType)[0] == -1){

	//FIXME: Register not available
	return;
}

if(get_modbus_register(deviceNum, registerAddress, registerType)[0] == 0){
	valueToSet = 1;
}
//printstr("VALUE TO SET");
//printnum(valueToSet);

set_modbus_register(deviceNum, registerAddress, registerType, valueToSet); //Sets modbus register
for(var i = 0; i < 10 && get_modbus_register(deviceNum, registerAddress, registerType)[0] != valueToSet;
i++){
	sleep(1);
} //Waits 10 sec for register to be set

if(get_modbus_register(deviceNum, registerAddress, registerType)[0] != valueToSet){
	//FIXME: Did not set value, maybe device is not available
}
```

3.5. Save sonos state, set volume to 20, say temperature outside and restore the state

```
send_udp("127.0.0.1", 5010, "sonos;;snapshot;;My zone");
send_udp("127.0.0.1", 5010, "sonos;;volume;;My zone;;20");
send_udp("127.0.0.1", 5010, "sonos;;say;;My zone;;Hey, today is " +
Integer.parseInt(get_weather_temp()) + " degrees Celsius;;en");
send_udp("127.0.0.1", 5010, "sonos;;restore;;My zone");
```

4. Version and changelog

1.0 – Initial version
1.1 – Added new examples
1.2 – Added get_temperature_room function
1.3 – Modbus commands added
1.3.1 – From version 2017-02-01 printnum, printstr, printfloat works only in testing mode
1.3.2 – Added sonos commands for new modules
1.4 – Added notification command