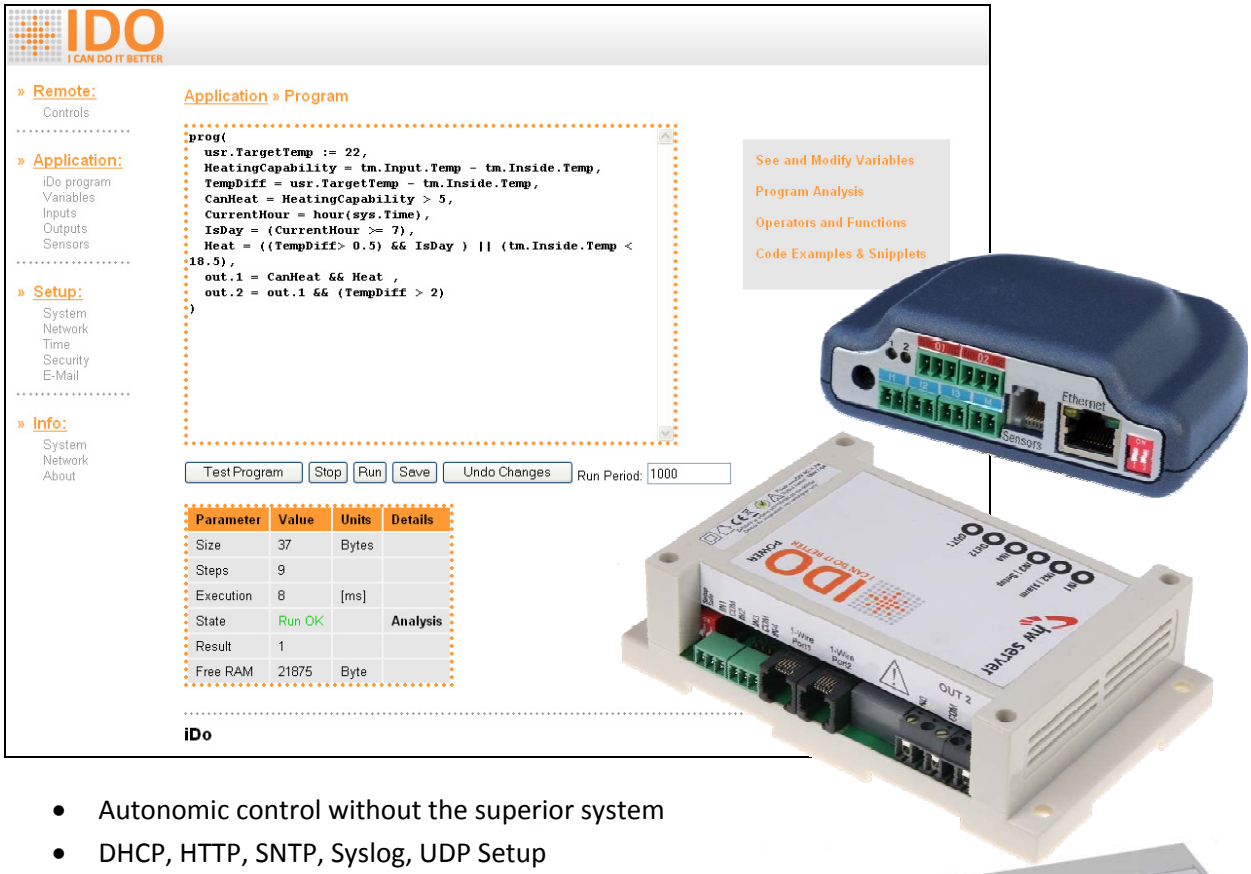
IDO

I CAN DO IT BETTER

Smart automation for everyone



- Autonomic control without the superior system
- DHCP, HTTP, SNTP, Syslog, UDP Setup
- XML, ASP
- Up to 32 1W sensors for temperature, humidity, voltage, current, pH, Redox, …
- Up to 16 logical inputs
- Up to 4 relay switching outputs
- Integrated development environment
- Easy to understand program language
- Hardware Watchdog
- Option custom user interface (XSLT, HTML, CSS, …)
- E-mail

Contents

Introduction:

Suggest you have a technology (electric light, water boiler, garage gate, or even floor convector or swimming pool) which you need automated control for ideally with network supervision and there is no suitable solution or it is not for reasonable price?

Have you ever given up some option just because development of such specialized tool for little series or even a single installation would be too expensive?

Have you ever resigned to a solution that was far from perfect, but the only one available on market for a fair price?

Are you interested in the energy wasting problem?

Do you want to make your house or flat more secure?

Do you need to integrate your household or company electronics into a single network?

If “yes” is an answer for these questions then iDo system is just for you.

What does it do:

- Lets you explain all tasks simply in your web browser
- Is able to communicate via Ethernet network (http, sntp, syslog)
- Is able to measure rigorous values (based on physical, electric and non-electric quantity) up to 32 sensors at once
- Is able to watch up to 16 logical inputs (buttons, contacts, switches…)
- Is able to control 4 switch contacts
- Is able to send measured as well as calculated values by XML
- UDP Setup
- Web upgrade firmware
- Is able send e-mail (SMTP compatible) based on program result
- Is able show info on external display unit (up to 127 rows)
- Support for comments inside program code (incl. multiline)
- Is able communicate over serial line with variable bitrates (iDo Pro only)
- Up to 32 1W devices (iDo Pro only)
- User defined XML
- WWW and XML documents stored in independent SPI memory, outside firmware code
- ASP support
- One code for every iDo type (differences are defined during compiling inside table of gates)
- Full support for AI sensors (A/D converters from HW Group)
- Optionally file browser and support for user customization
- Quadratic interpolation
- Linear interpolation
- Graphic display support (incl. fonts, bit-blitter, scenario and so on) (DB version only)
- Charon II development board distribution support

What does it maintain:

- Stair automats
- Lights
- Air-pumps
- Floor heating convectors
- Garage gates
- Swimming pool technology
- Whirlpools
- Straight heating on TUV
- Process time counters of welding apparatuses
- Distant monitoring of boiler rooms

Currently available types of sensors

- Multi-purpose galvanic isolated A/D converter for voltage and current. Easy setup for different applications
- Thermal 9bit
- Thermal 12bit
- iButton, iButton 1k, iButton 4k
- Relative humidity
- pH
- Redox
- Conductance

Versions

Version	1W BUS	Max sensors	Inputs	Outputs	RS232	Max relay current
iDo Net	1	8	4	2	0	1A
iDo Power	2	16	4	2	0	16A
iDo Pro	4	32	16	4	1	6A
iDo Control	2	16	12	8	0	500mA
iDo DB	1	8	1	0	2	-

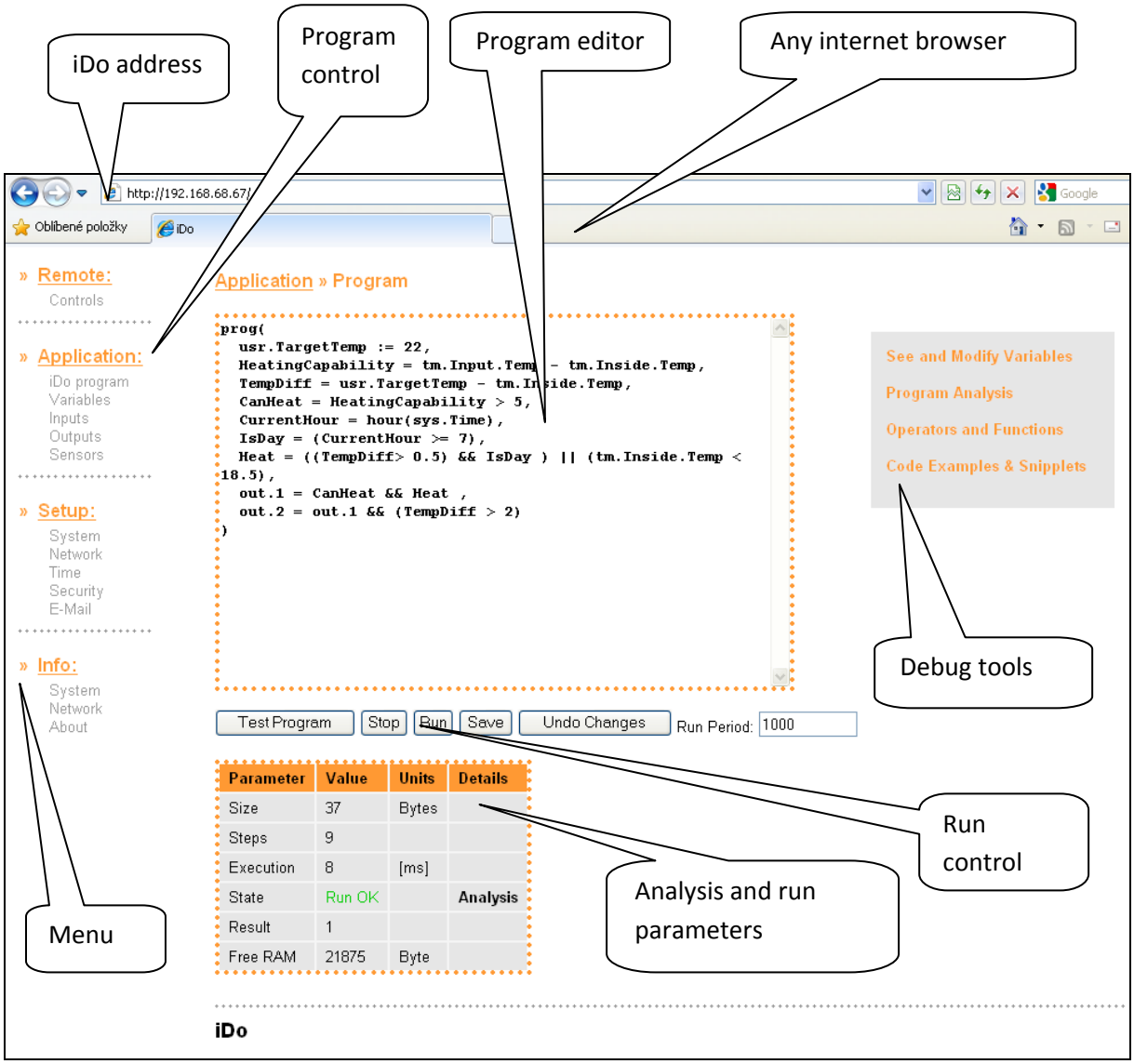
Why iDo?

Because it provides an interface for development and application of solutions, that are unique for each customer in the area of control and automation including its integration into a single network for a very reasonable price.

Because you can do it better.

Starters guide

iDo is able to memorize program defining actions of outputs based on the status of inputs, temperatures and values of variables. Program is defined, tested and maintained by common web browser. No additional tools required.



Everything necessary for creation, compile, debugging and running the program is part of your iDo device.

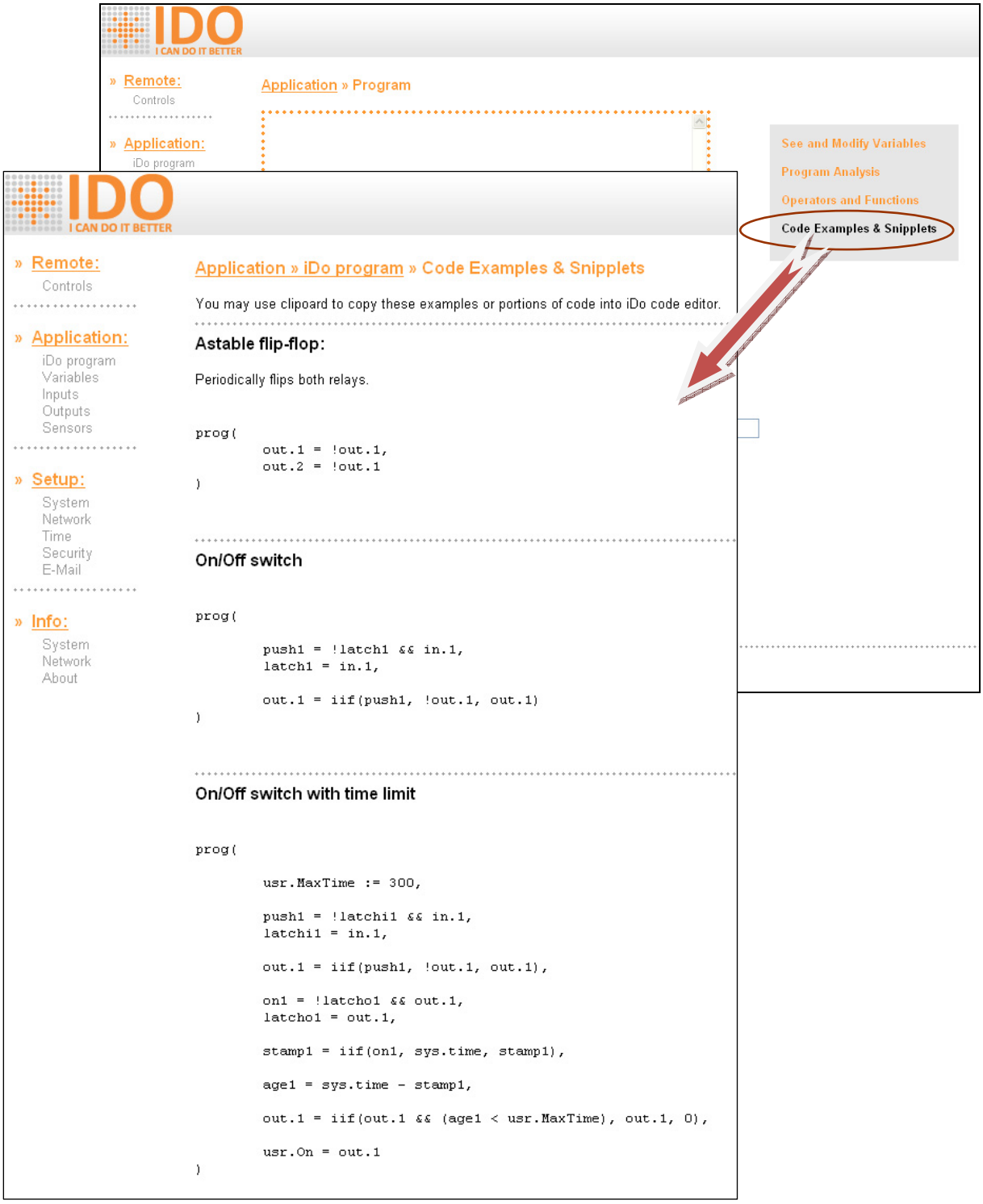
How to write a program

1. Open any web browser
2. Write address of iDo device into address field
3. Click “iDo program” in menu on the left side
4. Write program itself into editor window (you can even copy it from examples)
5. Click the Test button (on the bottom of editor window) to check correctness of your program
6. Click the Run button to make your program run
7. If you are satisfied with your iDo program, save it clicking the Save button

And that is all from now on your program maintains iDo functions.

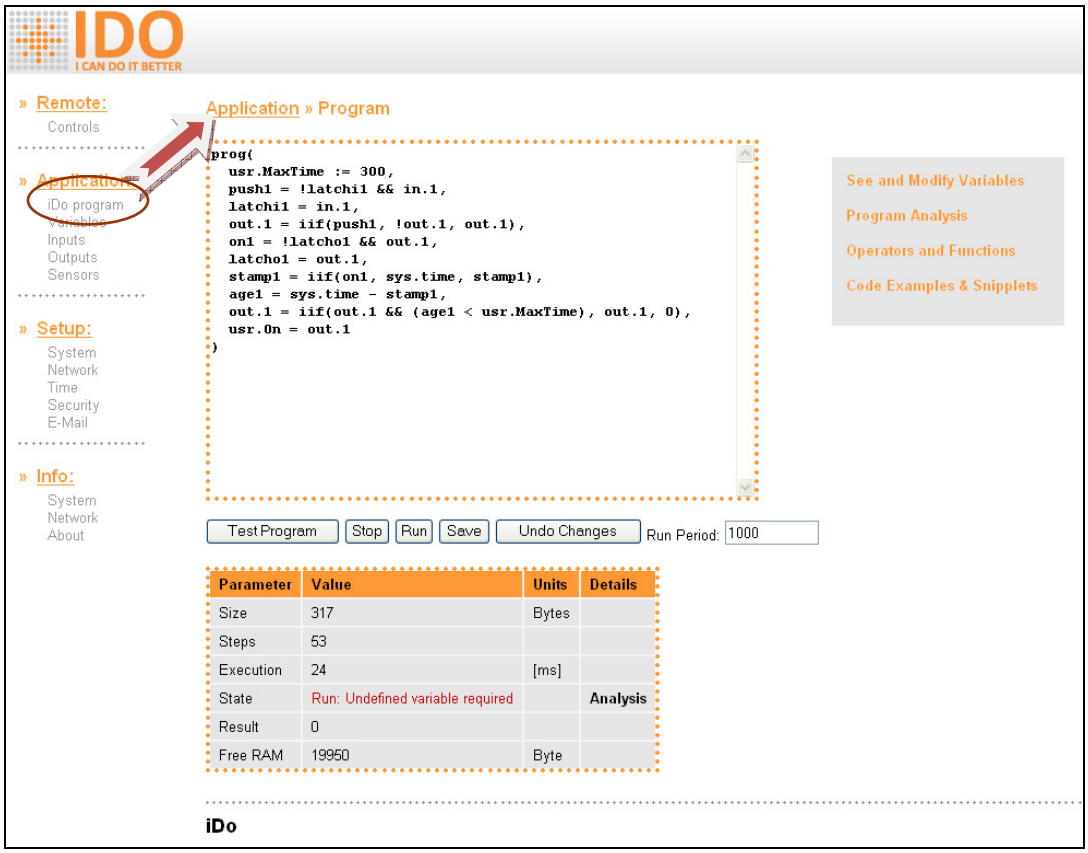
Parts and examples of a code

iDo contains applicable parts and examples of code which can be used as inspiration or even copied into your own program.

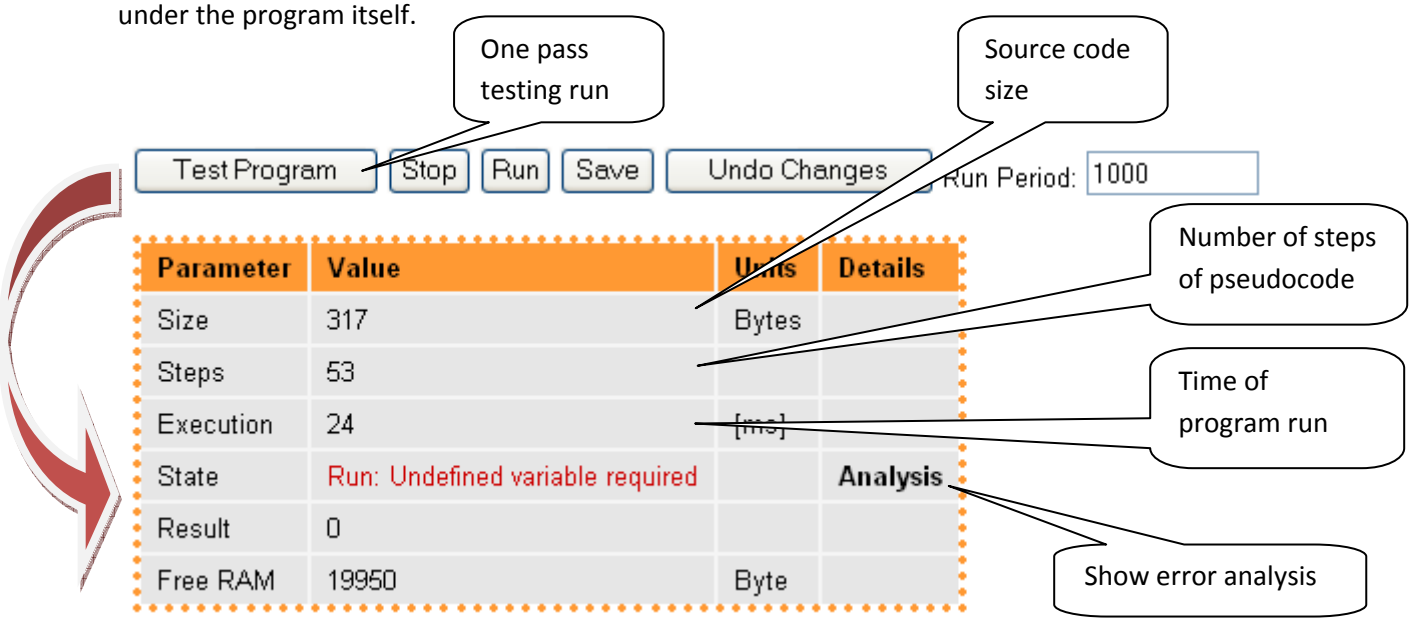


How to debug it

While making the program you may (and it is quite probable that you will) want to check whether your program is written correctly and does what it is supposed to do. For these occasions there are debugging tools as a part of your iDo device.

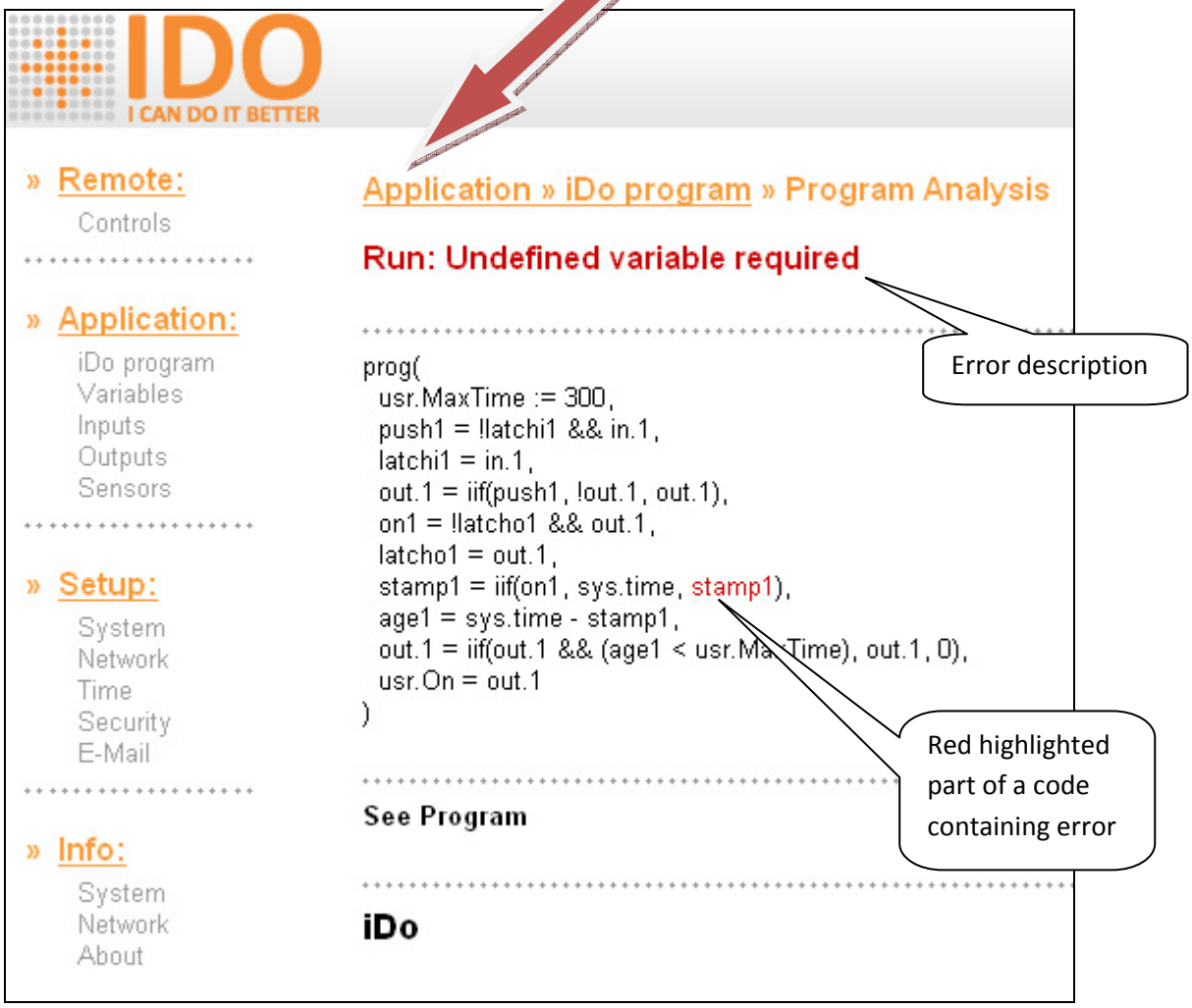


If you have a program in your editor window that is ready for debugging, just click the Test button. Program will be immediately compiled and launched once as a test run. Results will appear in table under the program itself.



If you want to know where the error is click on “Analysis” link, error will be highlighted with red color.

Parameter	Value	Units	Details
Size	317	Bytes	
Steps	53		
Execution	24	[ms]	
State	Run: Undefined variable required		Analysis
Result	0		
Free RAM	19950	Byte	



In case of a compile error everything from error on is highlighted, because the rest of program cannot be resolved correctly. Compile error even restricts test run of program.

After program is successfully compiled, test run follows. This run indicates all run-time errors. If error is unfixable, then process stops on error part which is highlighted in analysis. If the error can be fixed (for example undefined variable) then it continues and only last appearance is highlighted in analysis.

Error reports

iDo recognizes following errors:

Compile errors

Compiler identifies only obvious errors in program syntax; it does not check for example number of parameters needs for operators and functions, these errors are not identified until the test run.

Error	Description
Compile: Unrecognized token	Unrecognized token, compiler found part of the code that is not able to identify as a constant, variable, operator or function.
Compile: Expected function or operator	Program expects certain function or operator which cannot be found or program unexpectedly ends.
Compile: Unmatched bracket	There is a right bracket without the left one.

Run-time errors

Some errors, that does not occur until the test run can be result of mistake in program structure, these are considered very important and run is immediately ended. Other errors like calling undefined variable, undefined result of mathematical operation etc. are considered less important and program runs until the end. Depending on the nature of error missed value is replaced by 0 (for example in case of missing variable) or by NAN (Not A Number) value (for example in case of dividing by 0). Program analyzer highlights the last appearance of this error.

If program finds any error during its run, the results of program does not applied on physical outputs (for example out. 1, out. 2, LED etc.).

Error	Importance	Description
Run: Value expected but none found on the stack	High	Operator or function demands missing operand or argument, this is error of program structure
Run: Symbol expected but none found on the stack	High	Operator or function demands missing name of variable, this is error of program structure
Run: Function instead of value	High	Operator or function demands value and there is function or operator instead of it, this is error of program structure
Run: Function instead of symbol	High	Operator or function demands name of variable and there is function or operator instead of it, this is error of program structure
Run: Missing token with value	High	Operator or function demands value, this is error of program structure
Run: Missing token with variable	High	Operator or function demands variable, this is error of program structure
Run: Undefined variable required	Low	Undefined variable is demanded, 0 inducted
Run: Invalid variable name	High	Name of variable is invalid
Run: Not a Number	Low	Result of operation is not defined, NAN will be used instead

The error “Undefined variable required” can occurred in seemingly fault free program.
Example:

```
Prog(
      Next = Previous + 1,
      Previous = Next
)
```

This program will display error on first line at first run/pass. Variable “Previous” used on right side of expression is undeclared at this time. The variable is defined on second line, so second and all next pass will be OK.

If described compiler behavior is confusing for you, declare important variables in advance, using “:=” operator.
Example:

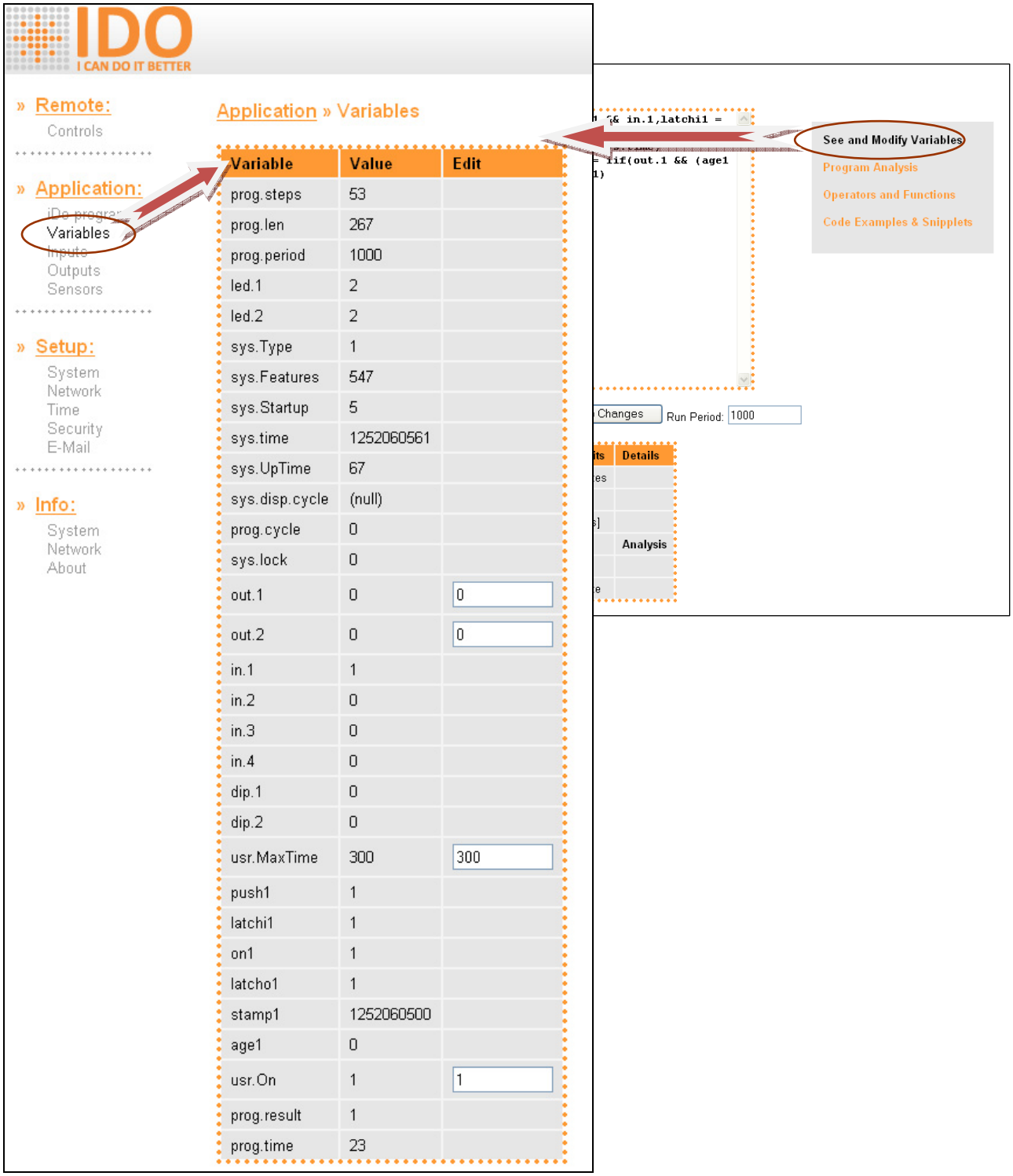
```
Prog(
      Previous:=0,          {If variable doesn’t defined, define it
and then assign default value}
      Next = Previous + 1,  {Calculate Next based on Previous}
      Previous = Next       {Store new value for Previous}
)
```

Program above is properness and will run without errors always.

Display of variables

If your program does not report any errors and still does not do what you expect it to do or in case you just want to take a look how your program works you can display list of variables including their values.

That is possible in main menu (Application > Variables) or straight from your program editor window (link “See and Modify Variables).



How to run a program

Once you have errorless program saved in iDo, it will run automatically in intervals you defined after you launch it until the device is turned off or program is stopped by user.

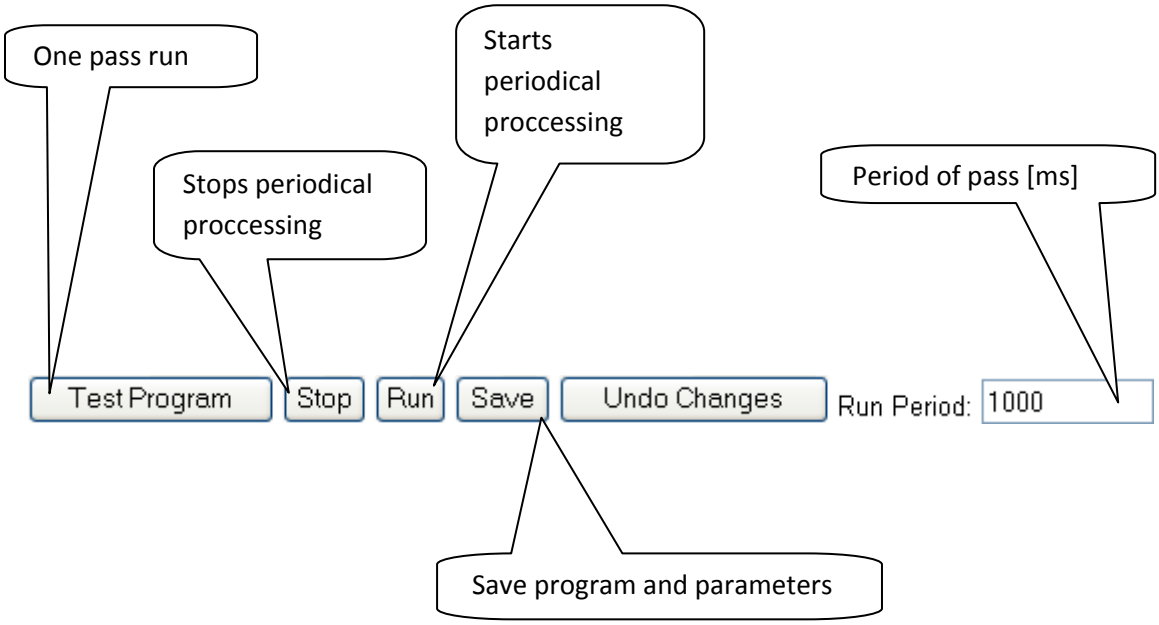
If an error occurs while program is running, there are two situations that may happen:

Error is serious	Program stops at error
Error is repairable	Program continues to the end

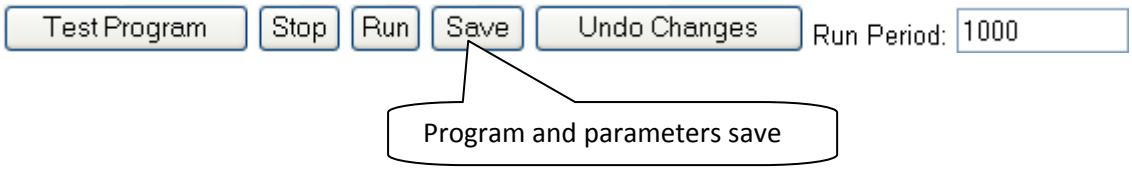
In both cases the results don't apply to the physical outputs. Outputs don't change, but value of variables does.

Even if error occurs program launches again in the next period. This prevents the program to stop in case of temporary error (for example sensor failure etc.)

During debugging or in other cases you can adjust program run by buttons underneath the program editor. You can also set process period (which means length of interval until then program is launched again).



How to save a program



Clicking save button you save program as well as run period.

Rules for program structure

There are following rules for programs:

1. Program consists of “prog” function which can have any number of arguments, separated by comma. Result of prog function is equal to the value of last argument.
2. Arguments consist of one or more mathematical/logical expressions.
3. Expressions consist of constants, variables, functions and operators.
4. Expressions are evaluated using standard rules of mathematical operations.
5. Each expression (including assignment) has a value which may be used in other operations.
6. Functions and operators are permanently defined in iDo device.
7. Access to inputs, outputs, thermometers, real time and user interface is realized by special variables maintained by system.
8. User can define his own variables.
9. Comments are enclosed in compound brackets, comments inside comments are not allowed.

Example:

```
Prog(out.1 = 1, out.2 = 0)
```

Turns on output relay 1 and turns off output relay 2.

```
Prog(
   out.1 = !out.1,
   out.2 = !out.1
)
```

Creates “blinker” with inverse outputs.

Variables

The name of variable can consist of any sequence of letters, numbers and a dot, as long as it begins with letter.

If program contains undefined variable, the zero value (0) is assigned while running, but the results ARE NOT APPLIED on physical outputs.

System variables

System defines following set of variables as default:

dip.1	Status of DIP switch 1	0 = off, 1 = on	Read only
dip.2	Status of DIP switch 2	0 = off, 1 = on	Read only
out.1	Status of output 1	0 = off, 1 = on	Read/write
out.x	Status of output x	0 = off, 1 = on	Read/write
in.1	Status of input 1	0 = off, 1 = on	Read only
in.2	Status of input 2	0 = off, 1 = on	Read only
in.3	Status of input 3	0 = off, 1 = on	Read only
in.x	Status of input x	Read only	Read only
sys.time	Real time	in seconds since 1/1/1970	Read only
prog.cycle	Process cycle of program	Process cycle of program	Read only
prog.result	Program result	Last program result	Read only
prog.time	Duration of program	[ms]	Read only
prog.steps	Instruction	The number of instructions in P-code	Read only
prog.len	Source code	The length of program	Read only
prog.period	Period of program execution	[ms]	Read only
led.1	Green LED	0 = off, 1 = on, 2 = blinks during run	Read/write
led.2	Red LED	0 = off, 1 = on, 2 = indicate error status	Read/write
led.x	Another LED	0 = off, 1 = on	Read/write

System sets variables atomically before every launch of program. If program runs without any error output variables are atomically rewritten into physical outputs after the end of run.

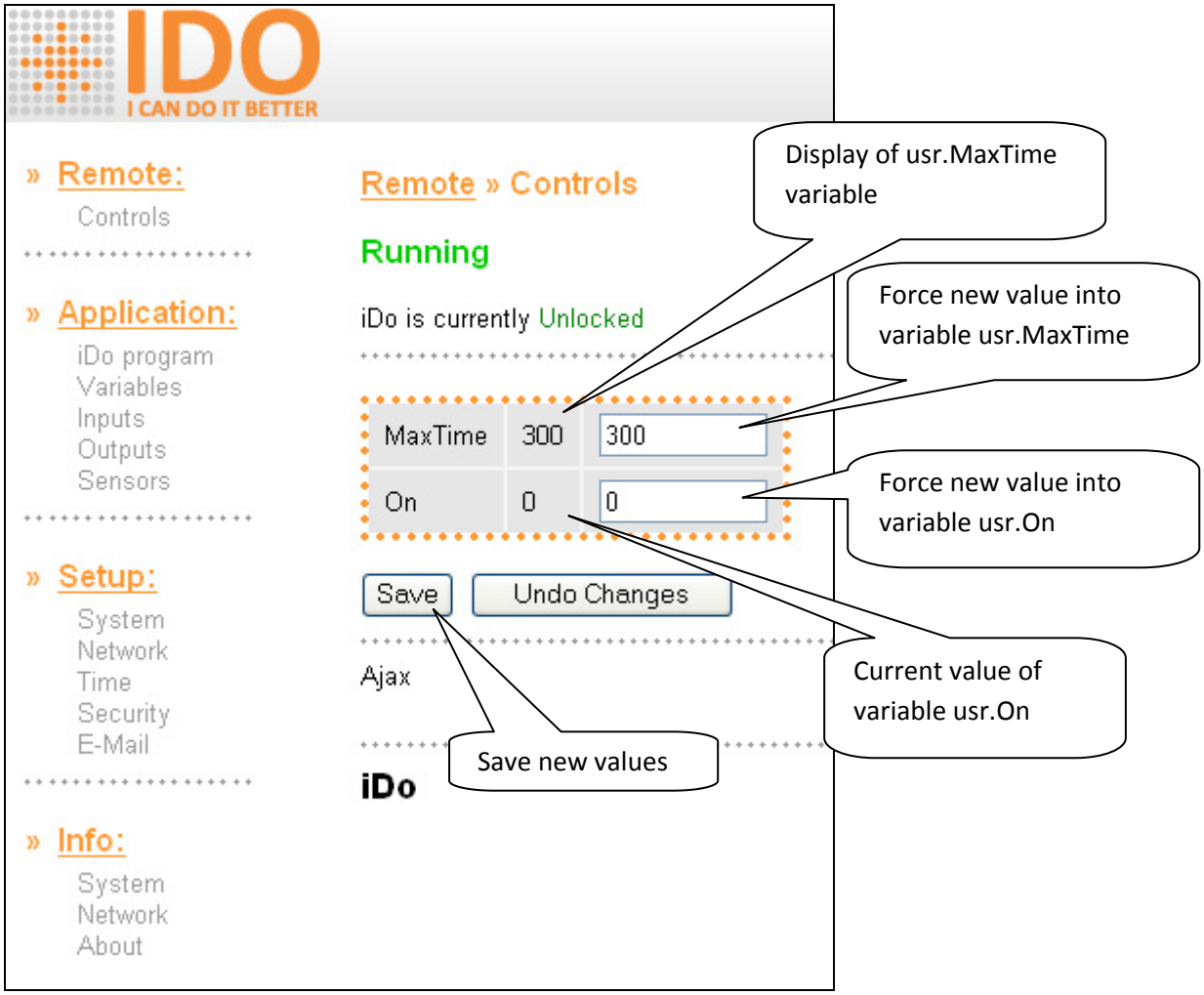
Program can change value of read only variables during the run-time, but assigned values are valid for current pass (cycle) only.

Control variables

Considering that almost every program has some value that controls its actions, for example desired temperature etc., iDo enables defining variables that are at disposal directly on homepage of device.

These variables are different only by its “usr.” prefix. It is recommended to define their value by operator “:=” so it cannot be rewritten during program run.

Default page of device:



Number of control variables is not explicitly limited.

Sensor variables

Sensor variables are always in the form.

`tm.name.property`

Where “tm.” is constant prefix of temperature sensors, “name” is a name (role, placement) of this sensor picked in table of sensors and “property” is desired property of sensor.

Names of particular sensors can be entered in table of sensors. The name of each sensor must complete following conditions:

- Length from 1 to 8 characters
- Only, letters, numbers and “_” (underscore) are allowed
- First character cannot be number

Properties of particular sensors are following:

Property	Value
Temp	Temperature in degrees of Celsius with accuracy 1/1000
State	Status of sensor
Change	Time of last change
Low	Lower limit set by user
High	Upper limit set by user

Appropriate variable is created only if it contains valid value.

If program contains undefined variable, its value is replaced with zero but results of program ARE NOT APPLIED on physical outputs

Example:

`tm.Inside.Temp`

The variable has value of temperature sensor with “Inside” name.

Variables from other units

iDo units communicate between themselves by Ethernet network and XML protocol. Implementation of mutual communication in your program is quite simple. All you have to do is to define variable which name starts as a name of unit you are interested in and continues by dot and a name of variable in distant unit.

For example:

```
out.1 = control.in.1
```

Causes that output 1 will copy status of input 1 of unit called “control”.

If you want to use this trick in your program first of all it is necessary to define variables that really interest you so to make our example work you have to make the whole program look for example like this:

```
prog(
   control.in.1 := 0,
   control.in.2 := 0,

   out.1 = control.in.1,
   out.2 = control.in.2

)
```

While on a side of distant iDo there is no necessity to set anything whole process is automatic and enables you to work with all defined variables which does not mean only system variables like inputs, outputs and sensors but also variables of running program.

Restrictions

Whole mechanism works under following conditions:

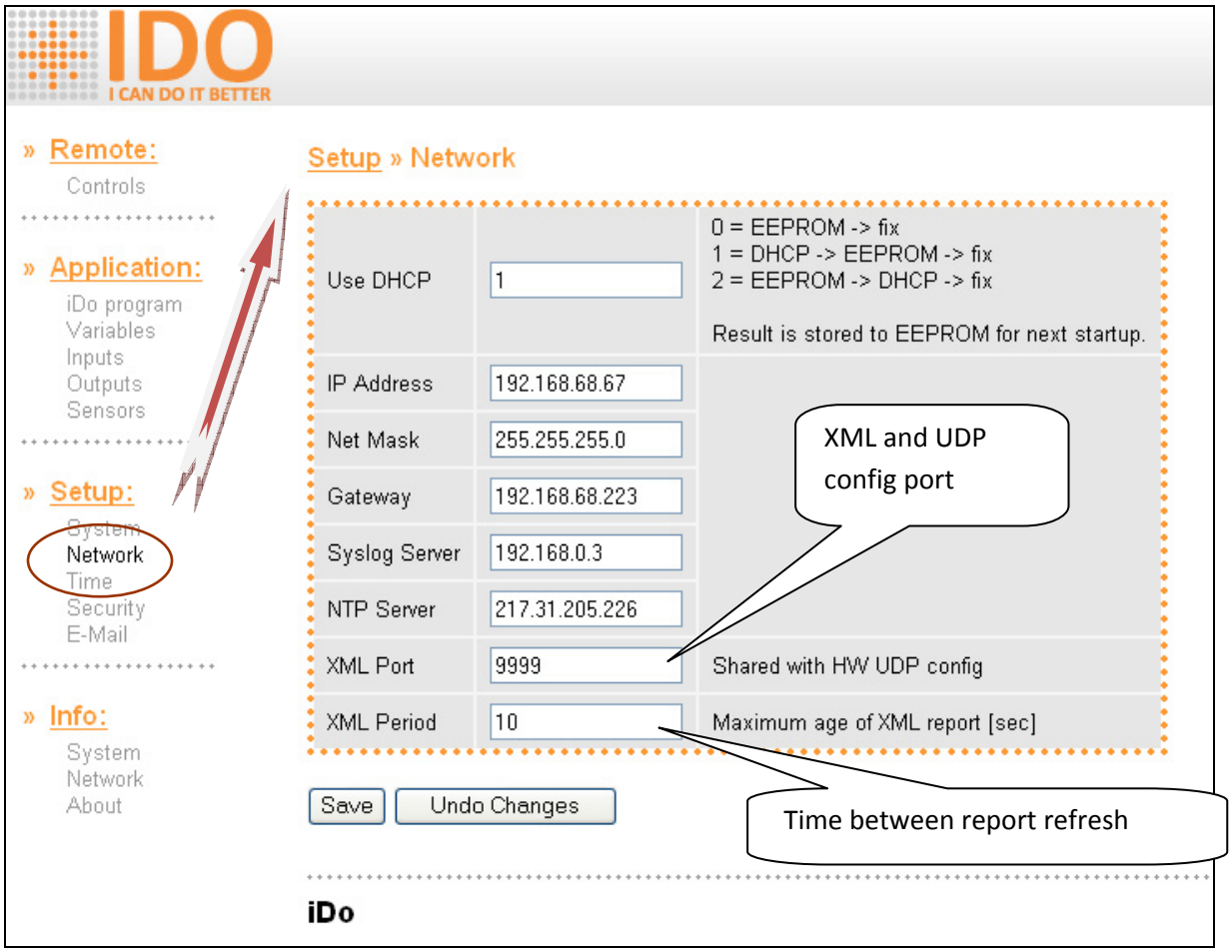
- Units are on a same subnet
- Units have valid address for this subnet (may be fixed or even assigned by DHCP)
- Name of shared variable does not exceed 16 characters
- Exchange XML file does not exceed 1500 B (size and structure of file can be checked on web address http://a.b.c.d/cgi-bin/device.xml

Communication

Mutual communication is provided by UDP protocol on port no. 9999. This port was chosen because it is already used by UDP Config function and units permanently listen on it. Units also use this port to report their status in regular periods or in case of important change. Whole mechanism is possible to disable on side of sending unit.

iDo contains sufficient XML parser which means, that this communication by sharing variables is not restricted for iDo units only. Variables can also be controlled by other applications; only thing necessary to keep is structure and placement of names and variables.

You can set port and period of XML communication at page Setup > Network:



Functions and operators

Programs of iDo device consist not only of variables and constants but also of functions and operators which are these:

Assignment operators

Operator	Precedence	Description
=	1	Assign (variable on left gets the value of expression on the right)
:=	1	Definition (if variable does not exist or has no valid value "Definition" works as "Assign" otherwise it does nothing.

Result of both operators is value of operand on right.

Arithmetic operators

Operator	Precedence	Description
+	10	Arithmetic addition
-	10	Arithmetic subtraction
*	11	Arithmetic multiplication
/	11	Arithmetic division
#	13	Arithmetic negation (unary minus)
%	11	Modulo (remainder of division)
div	11	Cuts out the part of number after the decimal point
pwr	12	Raise to a power of

All calculations with operators above are done with "double" accuracy. If operands are not of "double" type they are converted before calculation.

Example: 1 + 2 * 3 pwr 4 (result is 163)

Logical operators

Operator	Precedence	Description
!	13	Logical negation
||	2	Logical addition (or)
&&	3	Logical multiplication (and)

Logical operations are made using Boolean (true/false). If value of operand is equal to 0 (zero), or Not A Number (NAN), it is considered False. In all other cases its value is True.

Bit operators

Operator	Precedence	Description
|	4	Bit addition (bitwise or)
^	5	Bit exclusive addition (bitwise xor)
&	6	Bit multiplication (bitwise and)
<<	9	Bit move left (shl)
>>	9	Bit move right (shr)
~	13	Bit inversion

Bit operations works with “long” resolution. If operator has different type it is converted before operation.

Comparison operators

Operator	Precedence	Description
==	7	Test of equality, if operands are equal it returns 1 otherwise 0
!=	7	Test of inequality if operands are equal it returns 0 otherwise 1
<=	8	Less then or equal
>=	8	Greater then or equal
>	8	Greater then
<	8	Less then

Comparing is done with “double” accuracy. If operands are not of “double” type they are converted before calculation. Result is 1 or 0 value.

Determination function

Function	Arguments	Description
iif	3	If first argument has non-zero value than result is second argument otherwise result is third argument.

First of all determination function enumerates values of all arguments, afterwards it decides whether it returns value of second or third arguments based on value of first.

Converse functions

Function	Arguments	Description
int	1	Result is part of number before decimal point

Mathematical functions

Function	Arguments	Description
abs	1	Absolute value
min	2	Minimum
max	2	Maximum
exp	1	Exponent (e powered by x)
ln	1	Natural logarithm
sqrt	1	Square root

All calculations with these functions are done with “double” accuracy. If arguments are not of “double” type they are converted before calculation itself.

Timetable functions

Function	Arguments	Description
minute	1	Result is an hour[0,59] of time argument
hour	1	Result is an hour[0,23] of time argument
wday	1	Result is a day of a week[0,6](Sunday=0)

Argument of these functions is measure of time in “seconds since the Epoch” meaning seconds since midnight 1/1/1970, this time is also known as “Unix time”. Current value of this time is accessible by variable sys.time (assuming that your system time is set correctly)

Support functions

Function	Arguments	Description
prog	n	Result is the last argument

Function prog is technically pseudo-function its only purpose is to enclose whole program into logical structure a return system its result. Prog cannot be used in program more than once.

Support symbols

Symbol	Description
(	Left curved-bracket, adjusts priority of processing
)	Right curved-bracket, adjusts priority of processing
,	Comma, separator of arguments

In addition to these symbols anywhere in the program there can be any sequence of white spaces that has no influence on compile or run of program.

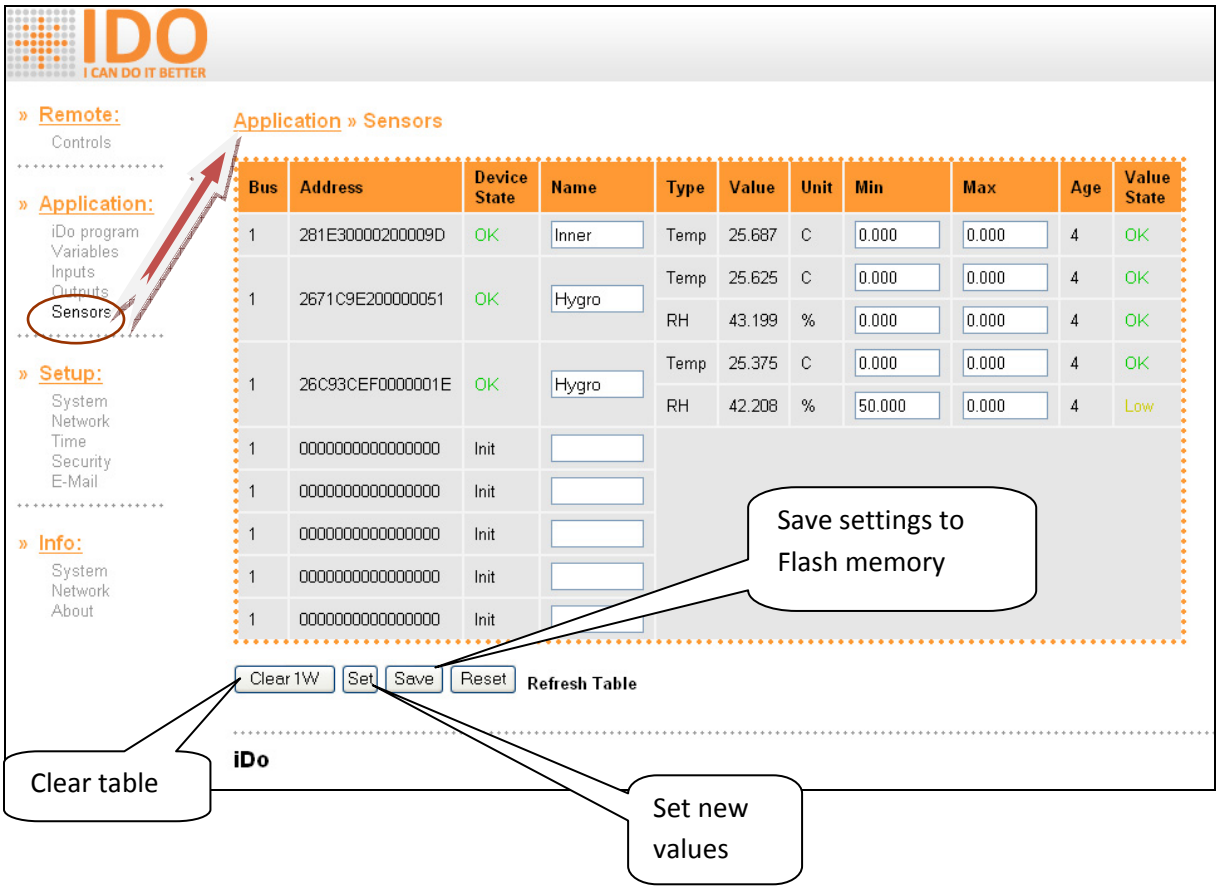
Current set of operators and functions installed on your device is at disposal in table version. See section Application > iDo program and link Operators and Function

Application » iDo program » Operators and Functions

Index	Operator	Precedence
1	(	-1
2	)	-1
3	,	-1
4	||	2
5	&&	3
6	|	4
7	^	5
8	&	6
9	==	7
10	!=	7
11	<=	8
12	>=	8
13	>	8
14	<	8
15	<<	9
16	>>	9
17	+	10
18	-	10
19	*	11
20	/	11
21	%	11

Sensors

Sensors are set using a simple table. iDo automatically scans 1W bus each 12sec. newly recognized sensors are immediately save into the table. If address of the sensor is the same as information saved in EEPROM predefined role is assigned to sensor.



Columns of table:

Column	Meaning
Bus	Number of bus at which is sensor attached (1 to 4).
Address	Address of sensor on 1W bus, unique identifier.
Type	Type of sensor (measured quantity). Type is obtained directly from the sensor.
Name	Assigned name, in some cases can be obtained directly from the sensor.
Device State	Hardware and bus status of sensor.
Value	Current value.
Unit	Unit of measured value. Unit is obtained directly from the sensor.
Low	Lower limit set by user
High	Upper limit set by user
Age	Age of last change
Value State	Current value state according to parameters Min and Max

The previous table of sensors creates following variables:

latch01	0	
stamp1	1252067100	
age1	0	
prog.result	0	
prog.time	19	
tm.Inner.Change	5	
tm.Inner.State	3	
tm.Inner.Temp	25.687000	
tm.Inner.Temp.High	0	
tm.Inner.Temp.Low	0	
tm.Inner.Temp.State	2	
tm.Hygro.Change	1252067093	
tm.Hygro.State	3	
tm.Hygro.Temp	25.281000	
tm.Hygro.Temp.High	0	
tm.Hygro.Temp.Low	0	
tm.Hygro.Temp.State	2	
tm.Hygro.RH	42.200001	
tm.Hygro.RH.High	0	
tm.Hygro.RH.Low	50	
tm.Hygro.RH.State	3	

Setting iDo parameters

iDo device allows users to modify wide range of parameters including behavior, page design and firmware update. Most of them are described in appendix 2 and 3. This chapter describes common user settings.

Restore company settings

Prior to change any setting is important to know how to restore unwanted mistake and recover default settings. To restore company settings and network configuration do following:

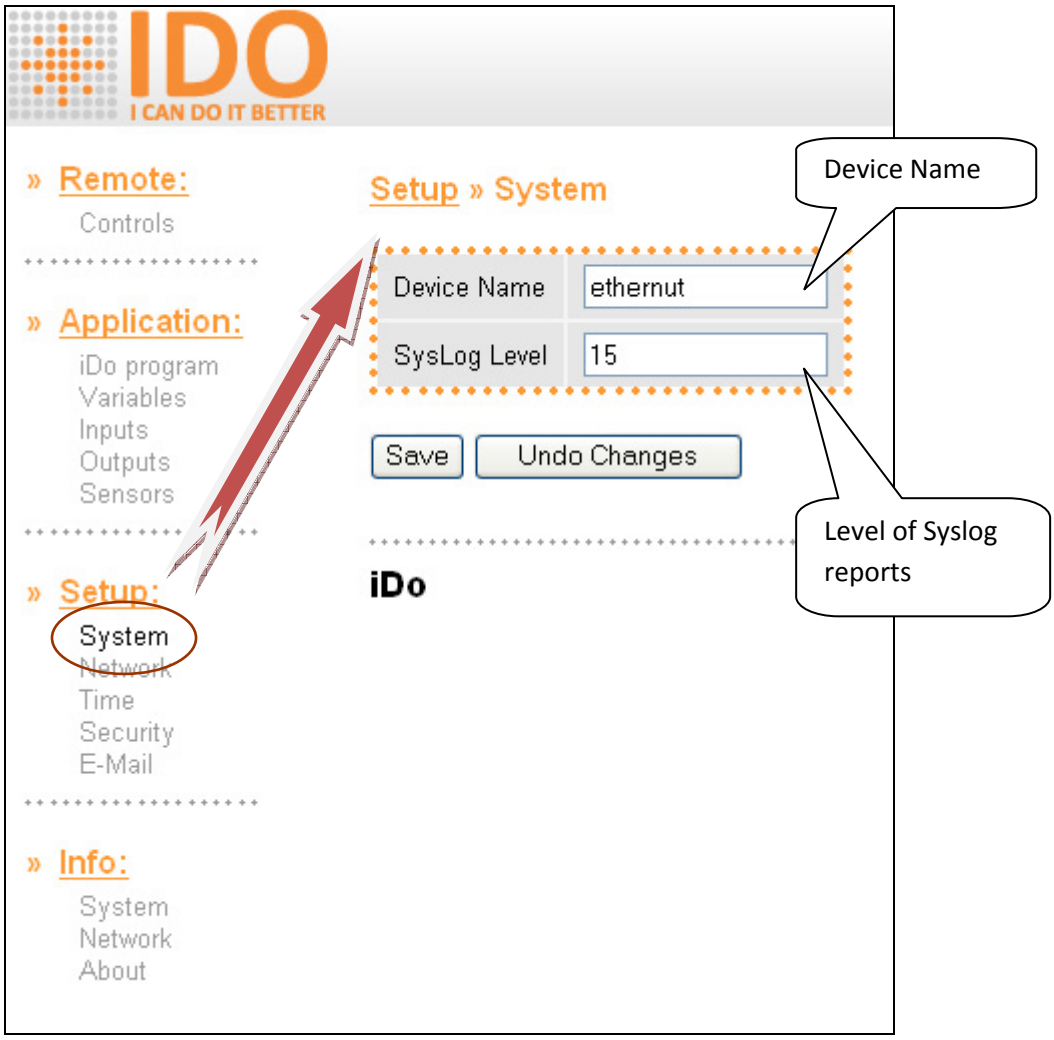
1. Switch off iDo device
2. Switch both DIP switches to position ON
3. Turn on iDo device
4. During following approx. 6 seconds will green LED blinks quickly. If during this period you switch both DIP switches to OFF position, the company profile will be restored.

Company settings use following values:

Parameter	Value	Meaning
DHCP	2	IP address is obtained in following order: EEPROM > DHCP > default IP (192.168.0.177)
IP	192.168.0.178	Device IP address (if DHCP failed)
MASK	255.255.255.0	Subnet mask (if DHCP failed)
GATEWAY	192.168.0.1	Gateway IP address (if DHCP failed)
SERVER	192.168.0.3	Syslog service server
NTP	217.31.205.226	NTP server

Using DIP switches in your application inflict this mechanism any way.

Setting application parameters



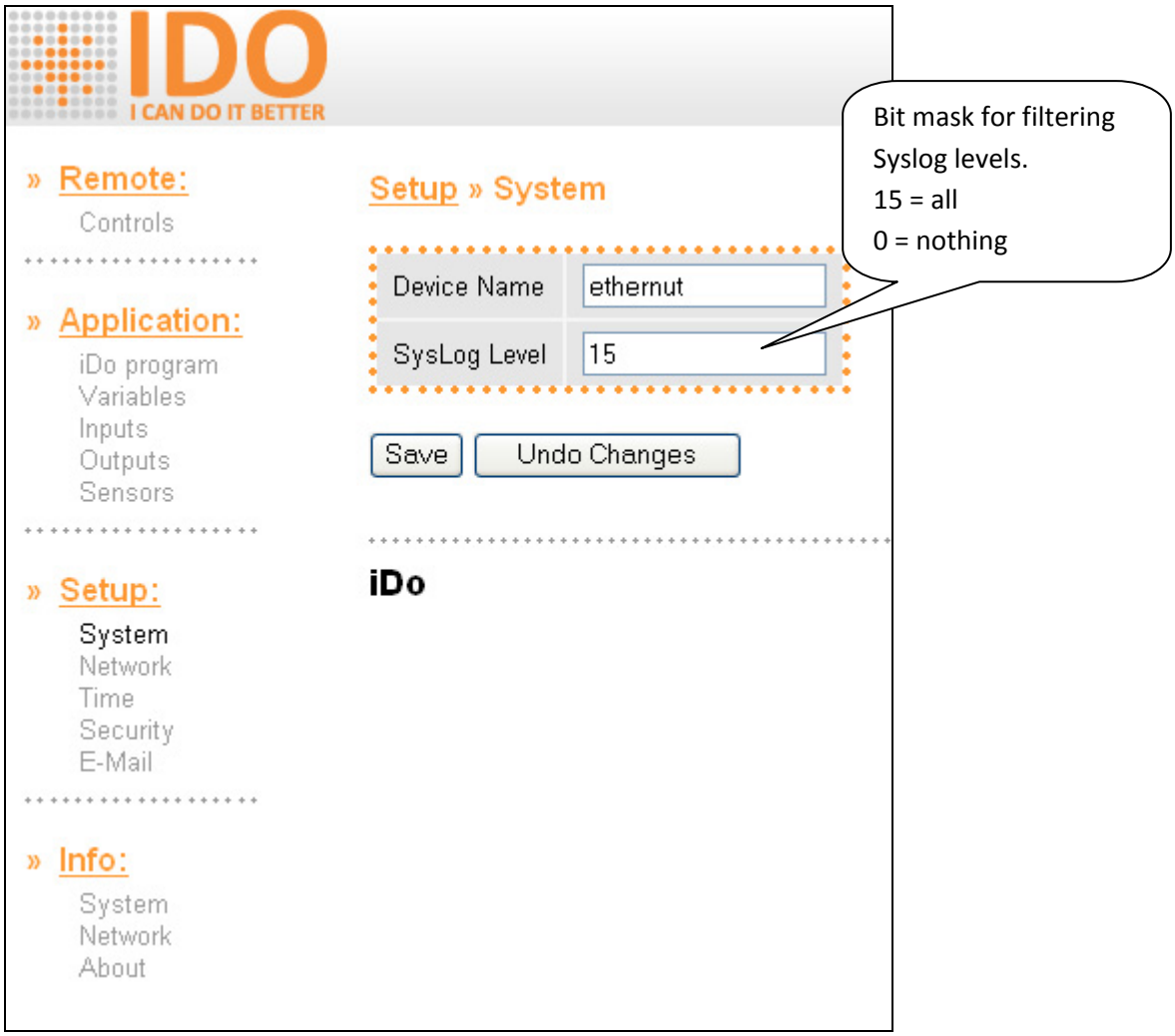
Syslog

iDo provides support of syslog system. Syslog is standard protocol for transfer of reports in IP networks. Detailed specification is for example in RFC 3164.

Reports are divided into following levels:

Level	Description
0	Emergency: system is unusable
1	Alert: action must be taken immediately
2	Critical: critical conditions
3	Error: error conditions
4	Warning: warning conditions
5	Notice: normal but significant condition
6	Informational: informational messages
7	Debug: debug-level messages

Outcoming messages can be filtered by setting SysLog Level on page Setup > System.



There are plenty of programs that can be used to catch and process Syslog. For freeware lets name for example Linux daemon syslogd (must be run with –r parameter) or Windows Tftpd32.exe.

Syslog events

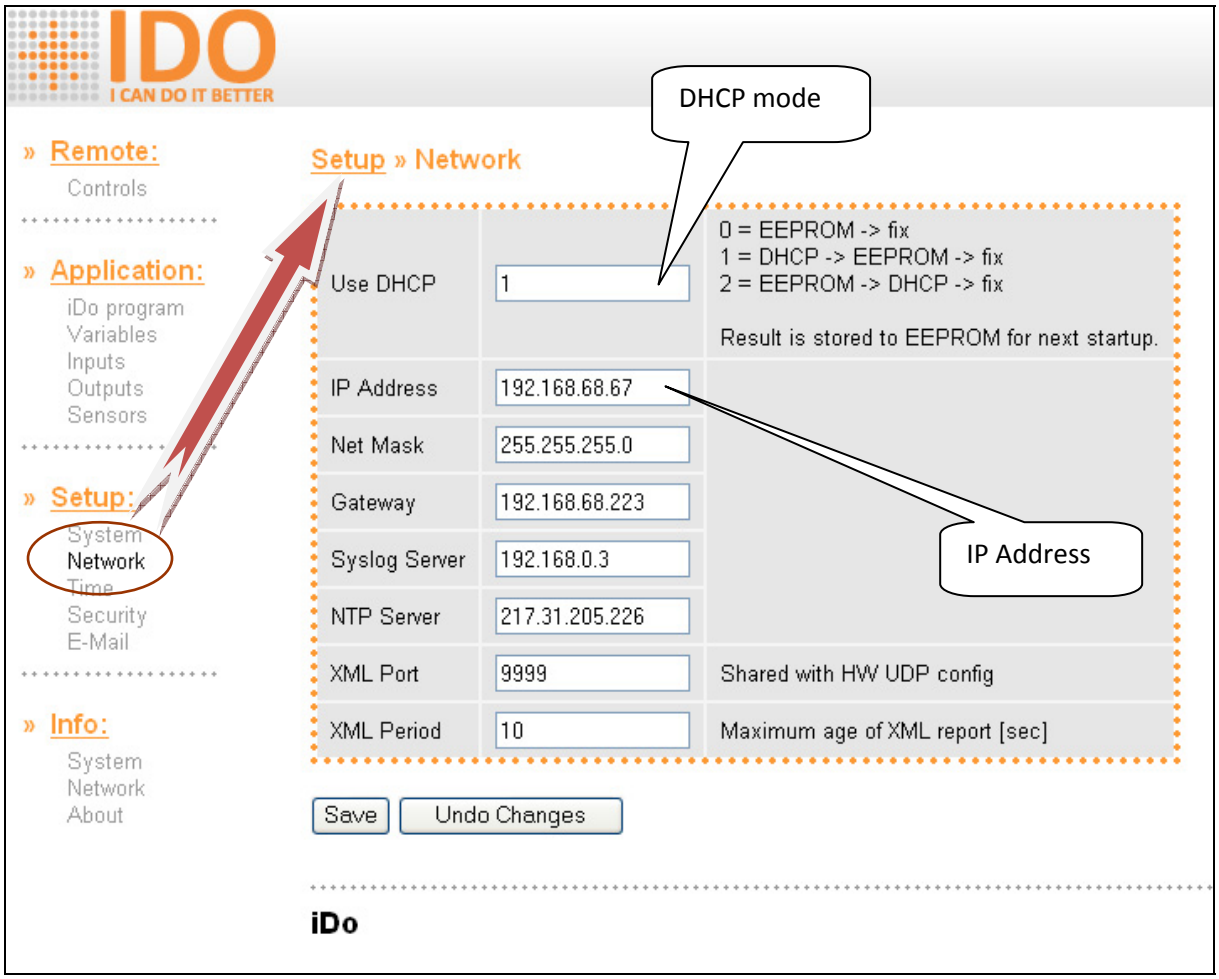
iDo can send reports in following situations:

Event	Level
System start	**Informational**
Periodical MARKER with 10min. period	**Informational**
Change of program result	**Notice**
Temperature out of setting limits	**Warning**
Error during program run	**Error**
Debugging report	**Debug**

Format of messages:

<time mark><address><name of device><type of device><firmware version>:
source;description;value

Setting network parameters

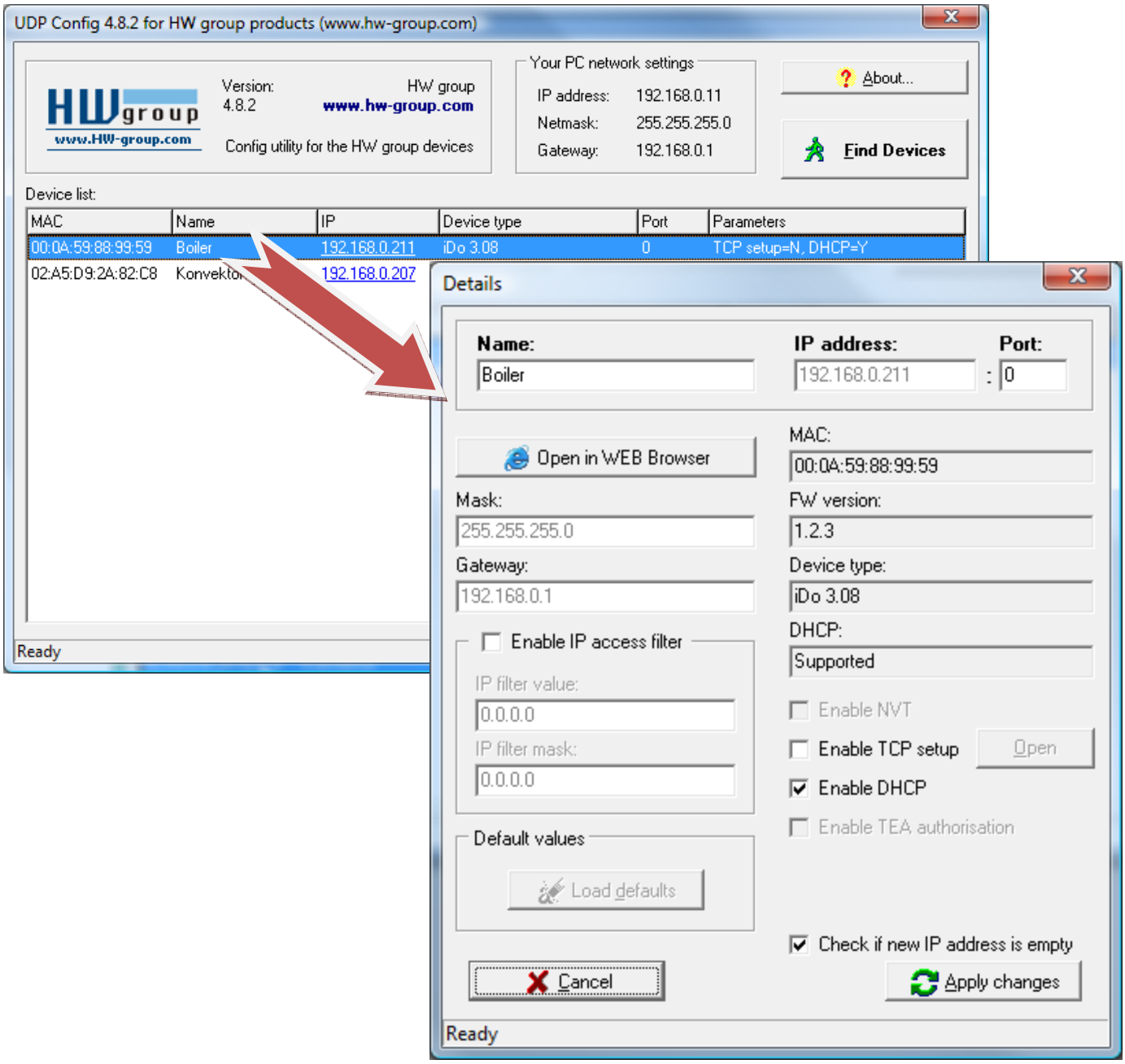


iDo has implemented support of DHCP. Default mode DHCP is 2 which means:

1. iDo checks if there is valid setting saved in EEPROM
2. If not it tries to get address from DHCP server
3. If address if not assigned “hardcoded” address is used (192.168.0.177)

If necessary you can find your device address (and eventually change it) using UDP Config application.

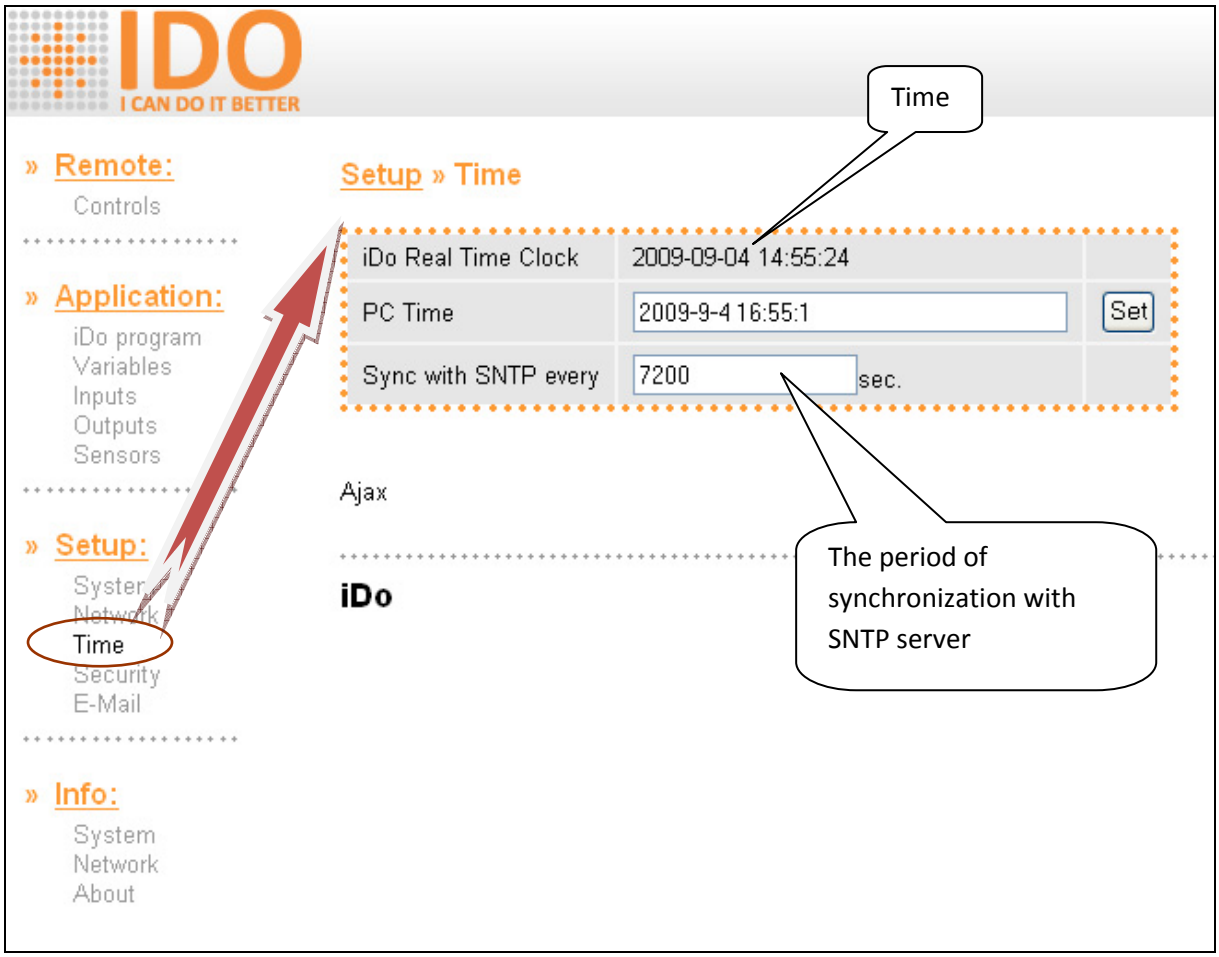
UDP Config



UDP Config finds all compatible devices and enables change of their setting and opening of their web pages.

Program UDP Config can be downloaded at address
http://www.hw-group.com/software/udp_config/index_cz.html

Real time setting



Security

iDo contains simple security solution to secure your settings. This mechanism is simple but totally fundamental security tool. It is absolutely necessary to secure your iDo connected over public network (e. g. internet) and same solution could be use in case when you need secure iDo settings on local network.

iDo knows two security state only:

- Lock
- Unlock

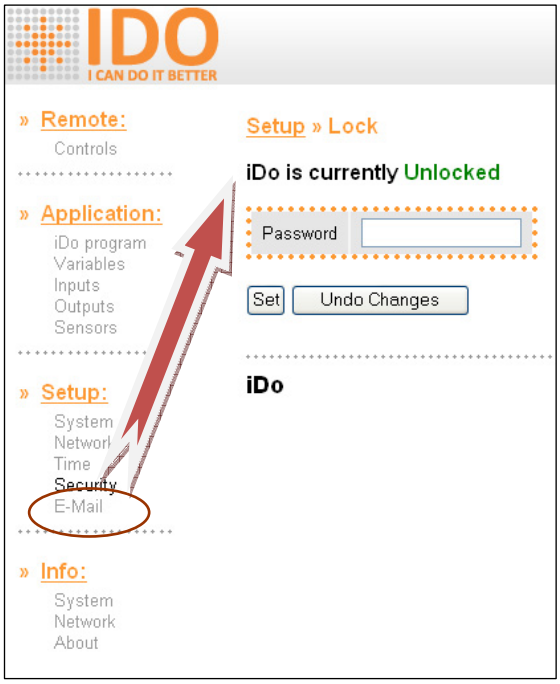
In unlock state every user can access any parameters, values and program too.

In lock state you cannot change following:

- Setting application parameters
- Setting network parameters
- Program
- Records in table of sensors

But all parameters above are available for reading, still.

Regardless of lock state, the changes of values of variables are allowed due to remote control functionality and communication between units.



How to lock

1) In left menu click item Security
2) Enter password into field Password
3) Click Set button

How to unlock

1) In left menu click item Security
2) Enter password into field Password
3) Click Set button

How to detect current state

The field with password is empty in unlock state.

In your program you can use variable:

`Sys.Lock`

Contains one of following values:

0	Unlock
1	Lock

E-mail

If you specify conditions, iDo can send you (or anybody else) e-mails when conditions are fulfilled.

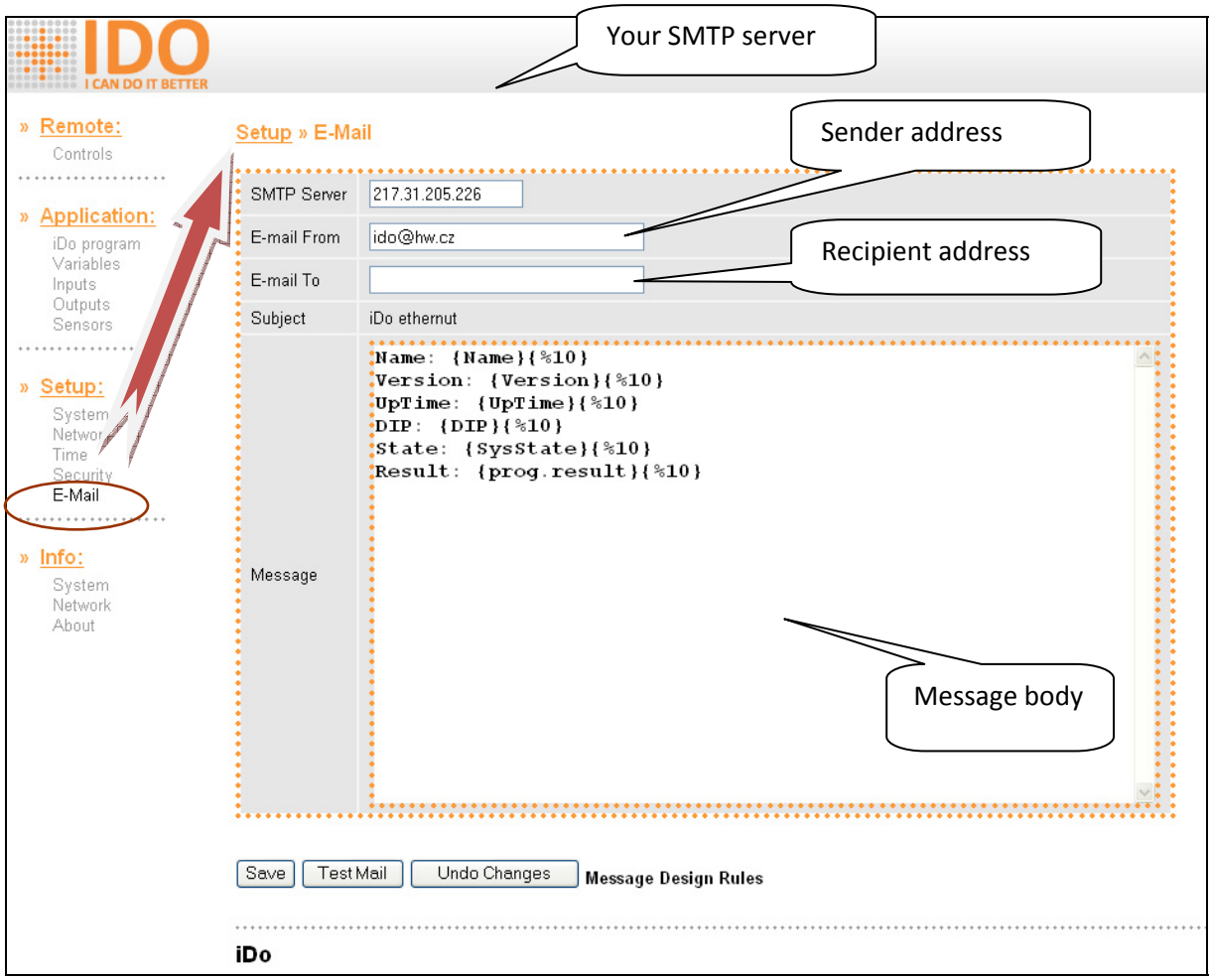
Condition is just one – change of program result.

For example program:

```
Prog(!(sys.UpTime & 7))
```

Sends e-mail every 8 seconds.

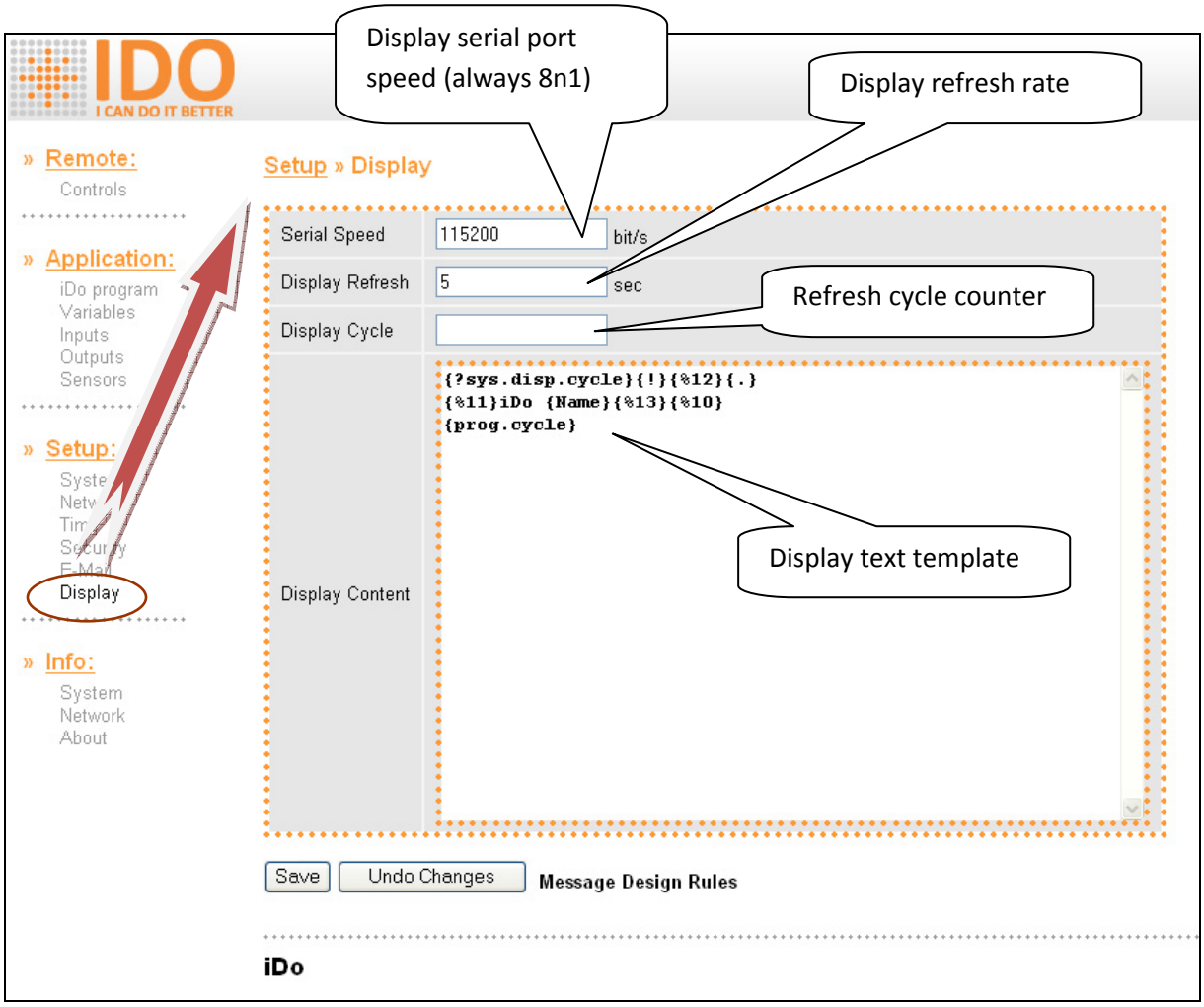
To work it all together it is necessary set following parameters:



You can use any variables in message body. See ASP files described in appendix 3.

Display

If your iDo have optionally serial interface (currently iDo Pro only) you may use external serial display unit. The parameters of display are available on page Setup ->Display:



Following displays are supported currently:

1. Generally any display units with RS232C interface and control characters for clear and home
2. iDo was tested with big green cashier display Posiflex PD2200
3. Next with LCD terminal from HW server company, see http://obchod.hw.cz/?cls=stoitem&stiid=36368
4. Next with serial TV printer, which is very impressive, call us for more info

How to control display

It's quite simple using template in field "Display Content". Template contains information displayed every time display refresh. Refresh rate is controlled by value in field "Display Refresh". Field "Display Cycle" contains counter incremented after each display refresh.

Example:

```
{?sys.disp.cycle}{!}{%12}{.}
{%11}iDo {Name}{%13}{%10}
{prog.cycle}
```

Firs row of template clears display after program run. Command Clear (Ascii 12) will commit when refresh counter is equal to zero.

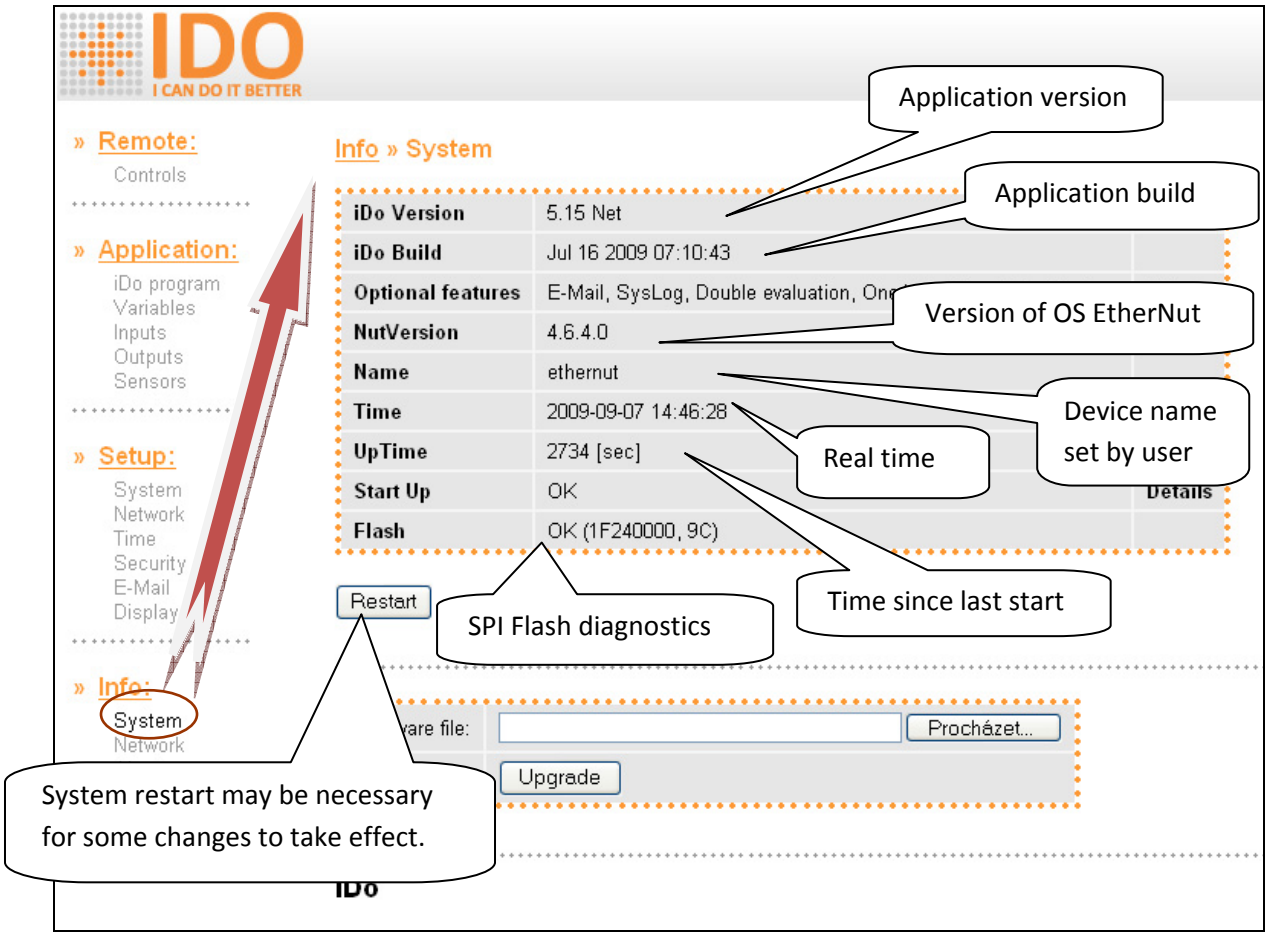
Second row of template moves display cursor to default (Home) position (Ascii 11). Text "iDo" will be displayed followed name of device (variable Name). Finally, the carriage return and form feed occur (Ascii 13, 10).

Last row displays current program pass counter (variable prog.cycle).

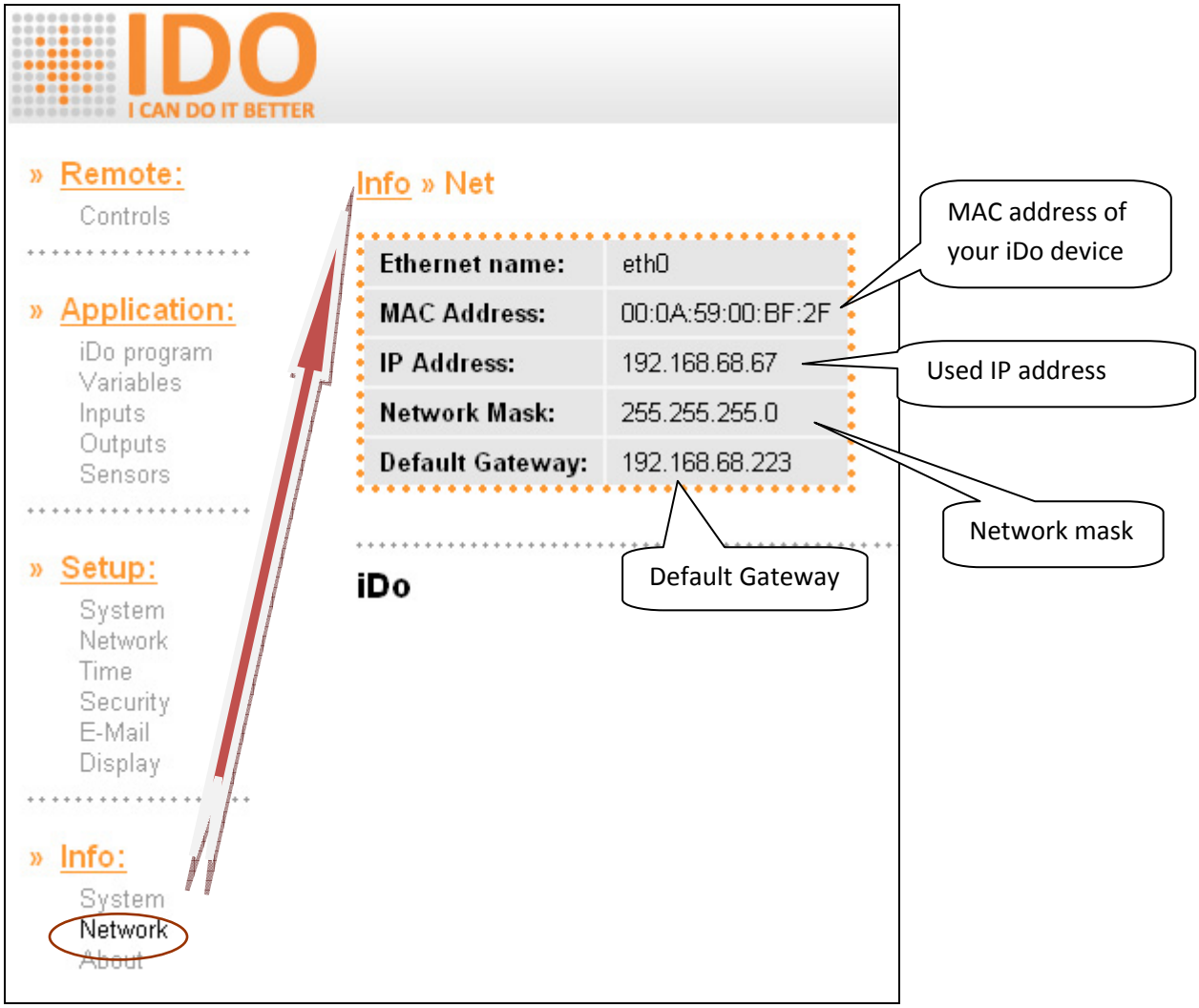
You may use any variables and parameters inside compound brackets in field "Display Content", see ASP files described in Appendix 3.

The link "Messages Design Rules" offers you brief overview of allowed commands too.

System information



Network information



About

And last but not least few useful links.

XML Interface

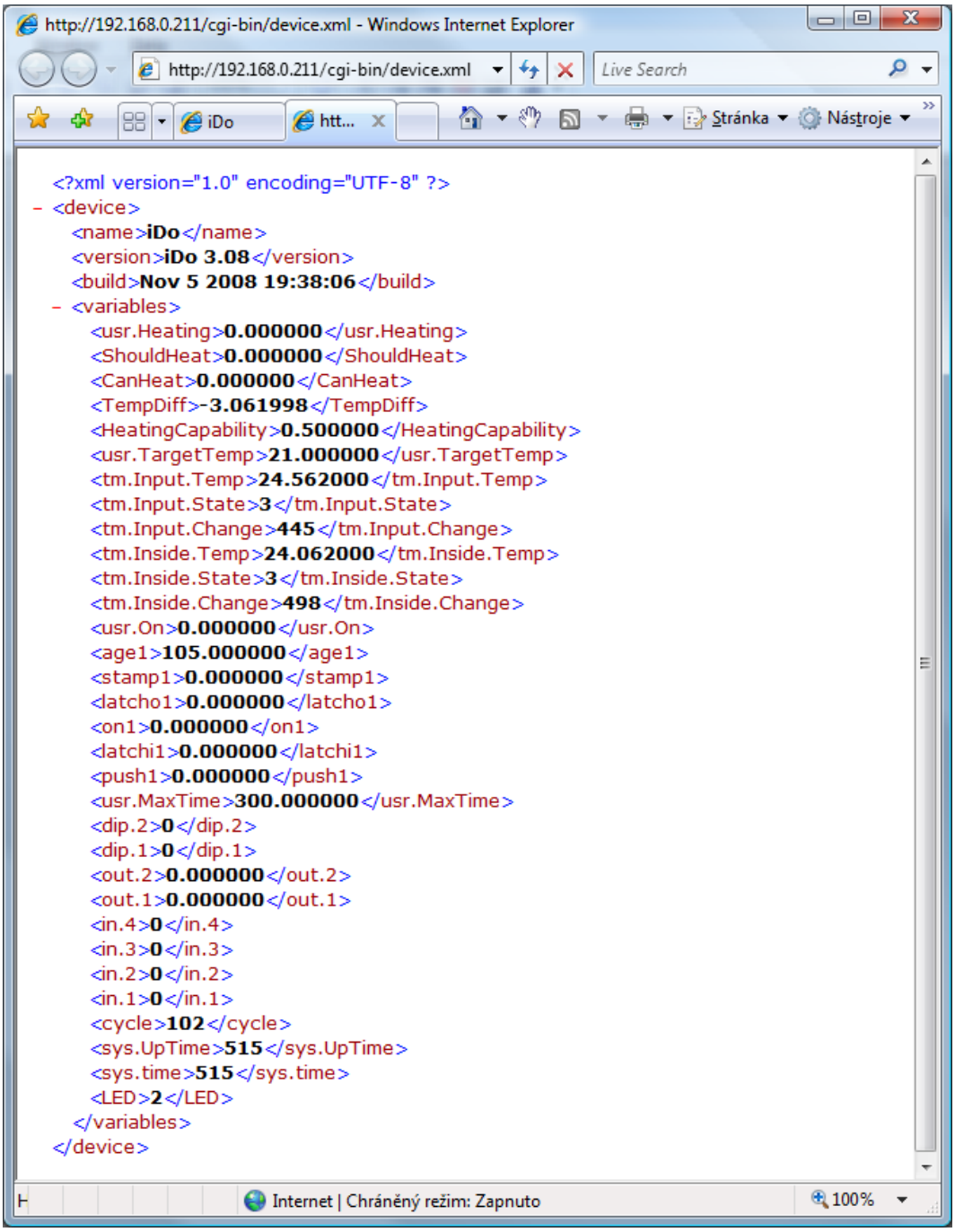
iDo has implemented XML interface enabling access to all program variables in real time, using network and superior control system.

XML data are transferred by http protocol and are available on:

http://address/cgi-bin/device.xml

Where address stands for address of your iDo device.

Data has following format:



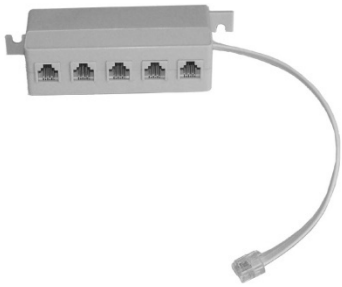

```xml
<?xml version="1.0" encoding="UTF-8" ?>
- <device>
  <name>iDo</name>
  <version>iDo 3.08</version>
  <build>Nov 5 2008 19:38:06</build>
  - <variables>
    <usr.Heating>0.000000</usr.Heating>
    <ShouldHeat>0.000000</ShouldHeat>
    <CanHeat>0.000000</CanHeat>
    <TempDiff>-3.061998</TempDiff>
    <HeatingCapability>0.500000</HeatingCapability>
    <usr.TargetTemp>21.000000</usr.TargetTemp>
    <tm.Input.Temp>24.562000</tm.Input.Temp>
    <tm.Input.State>3</tm.Input.State>
    <tm.Input.Change>445</tm.Input.Change>
    <tm.Inside.Temp>24.062000</tm.Inside.Temp>
    <tm.Inside.State>3</tm.Inside.State>
    <tm.Inside.Change>498</tm.Inside.Change>
    <usr.On>0.000000</usr.On>
    <age1>105.000000</age1>
    <stamp1>0.000000</stamp1>
    <latcho1>0.000000</latcho1>
    <on1>0.000000</on1>
    <latchi1>0.000000</latchi1>
    <push1>0.000000</push1>
    <usr.MaxTime>300.000000</usr.MaxTime>
    <dip.2>0</dip.2>
    <dip.1>0</dip.1>
    <out.2>0.000000</out.2>
    <out.1>0.000000</out.1>
    <in.4>0</in.4>
    <in.3>0</in.3>
    <in.2>0</in.2>
    <in.1>0</in.1>
    <cycle>102</cycle>
    <sys.UpTime>515</sys.UpTime>
    <sys.time>515</sys.time>
    <LED>2</LED>
  </variables>
</device>
```

Recommended accessories

iDo enables connection up to 32 sensors.

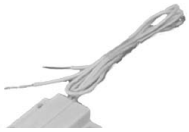
Sensors may be in indoor version.

For wet places and even extreme temperatures.



There are T-boxes and lengthening cables available for simple creation of sensor networks.

For logical inputs there are available for example:



- Flooding detector
- Smoke detector
- Detector of inflammable gases
- PIR movement detector
- Door contact

As power supply you may use:

- Standard power supply
- Backup power supply (UPS)

For strengthening outputs and capability to switch high power consuming devices:

PowerEgg – detector and controller of AC voltage from 110V up to 230V with galvanic separation. Input and output are low voltage contacts.

Appendix 1 – Real time management tasks

Intro

Despite accurate crystal controlled real-time clock inside iDo and capability to synchronize time with NTP server, the problems and limits may occur during realization same real-time applications. The main goal of this document is show you the solution of above.

Jumps back in time

Capability to synchronize time with NTP server is big advantage and also probably one possible solution to keep more devices synchronized. On other side please keep on your mind that time correction can run forward but in back direction too. When correction happens on minute border, the jumps back in time may occur in minute, hour or day dimension.

To prevent mentioned situation and unattended program behavior, we suggest setting NTP interval for time correction on once or twice time per day basis. For correction choose properly part of day, when time jump (few seconds) isn't critical for you application.

Program (Run) period

When you design program for specified time resolution, the run period must be less then required time resolution. For example, when you required time resolution 10 seconds, then run period must be less then 10000 ms, vice versa at 1 second resolution the run period must be less then 1000 ms.

Calculation accuracy

iDo use in calculation short double data type with 32bits resolution in flow point format. Short double is best suitable for calculation with decimal values gained from temperature, humidity and other sensors. Unfortunately, in case of big integer value the accuracy limits may occur. The inner representation of time is typical example. iDo store time as a count of seconds from midnight 1/1/1970. For example time of creation for this article is 1241269920, after convert to double format we obtain just 124126988, next change will occur at T=1241269951 on value 1241270016. Accuracy in time calculation is 128 seconds always, in other word approx. 2.13 minutes.

Because some application required more precision, selected functions were modified to work with resolution of 1 second from version 5.09. But to achieve more precision, you must exploit system variable only! For example:

Minute = minute(sys.time)	return correct minute always
Minute = minute(sys.time +1)	return value with accuracy 2.13 minutes

Improved precision for time operation affect from version 5.09 following functions:

- hour()
- minute()
- wday()

- age()
- bits()

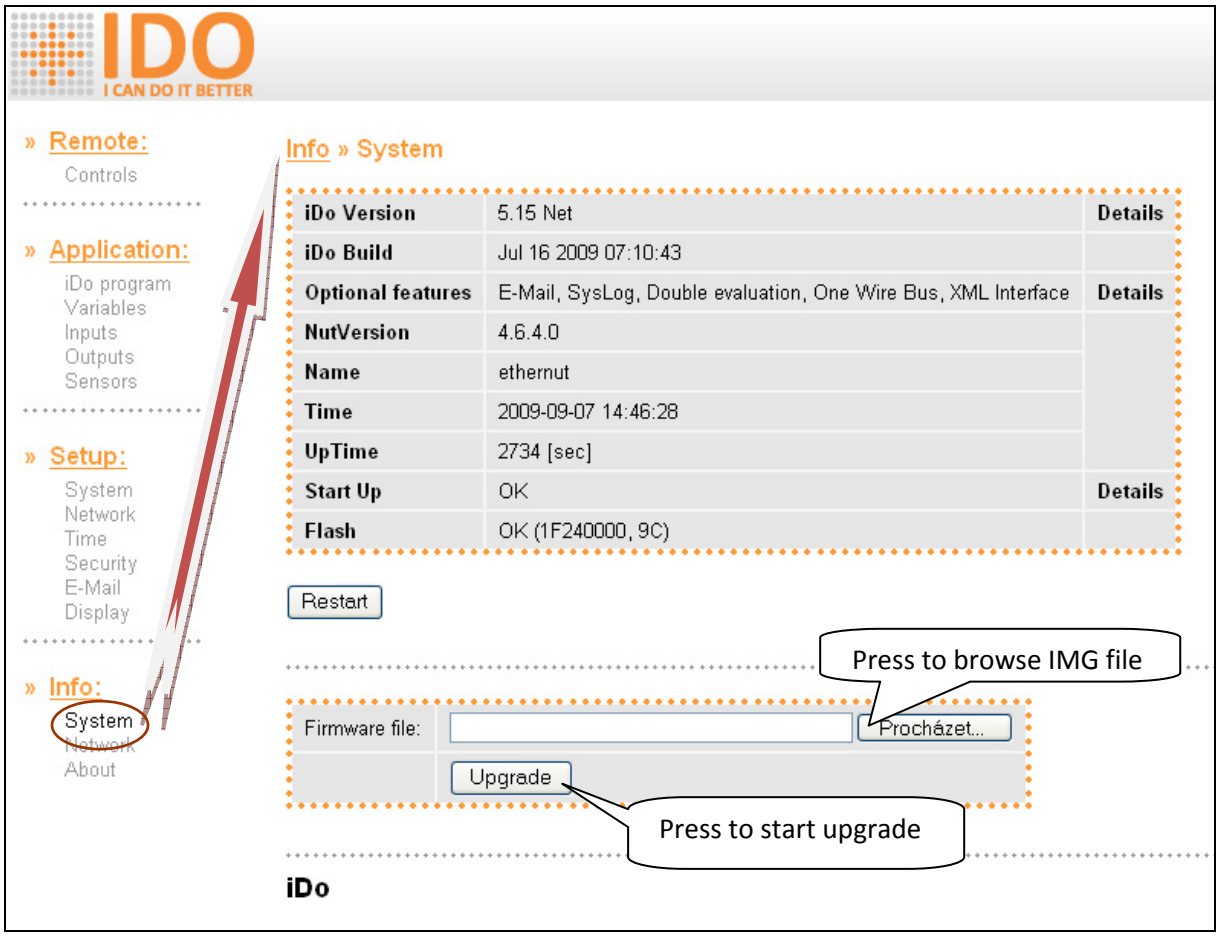
If you satisfied with low precision and you have no other reason for upgrade, we suggest you compare time value with different operator then “==”.

Full precision

If you build time-based application or you need from other reasons full precision for big number in your calculation, you may use firmware which use long type instead short double. This firmware you obtain after request.

Appendix 2 – Upgrade

iDo supports FlashBoot system which allow comfortable upgrade via web pages interface.



Upgrade consists of following steps:

1) Gain IMG file with firmware (usually from HW server web)

2) Select link System from section Info and visit page with system information

3) Use button “Browse” to locate file with firmware

4) Use button “Upgrade” to transfer file into iDo device and start upgrade

5) Wait approx. 1 minute till pages refresh to show new info

During upgrade all contents of SPI Flash memory will be erased! You lost following data after upgrade:

- User program
- 1W sensors definition and settings
- Stored logs

- Stored variables

When something wrong happens

If iDo behavior is undefined after upgrade, try following steps:

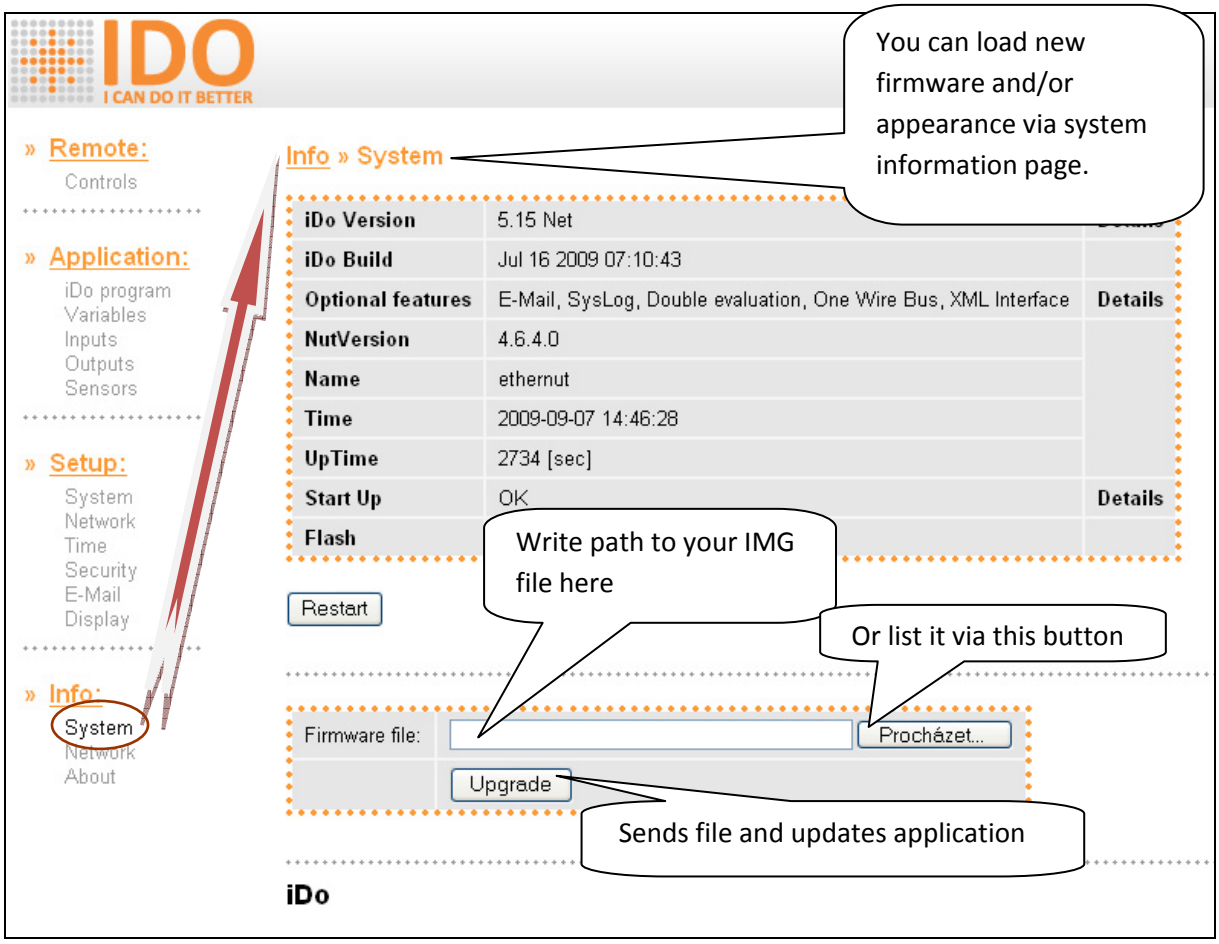
1. Check length of IMG file name, if exceed 15 chars bootloader cannot process it and upgrade fails. To solve this situation use links at end of the chapter.
2. Clear cache (temporary internet files) in your internet browser and force page refresh (actual information will be displayed instead).
3. If you flash firmware indeed for different type of iDo, just do new upgrade with correct firmware type. Sometimes you must switch off iDo for a moment prior new upgrade because different Ethernet initialization is used for some types of iDo devices.
4. The situation is quite simple if you flash something other (then IMG file). Generally, you are just overwrote SPI flash memory with senseless file. CPU content stays untouched if bootloader cannot find valid image. New upgrade is possible using links at end of the chapter.
5. If iDo doesn't answer after upgrade nor http protocol, UDP setup and Ping, the solution could be switch device off and on, but it doesn't happen often. This behavior signalized more often different kind of problems, e. g. power supply and etc.

Helpful links:

http://address/index.html	Show iDo home page if content of SPI flash memory is valid. If content of SPI flash memory is invalid, hard coded page will be displayed, which allow you upgrade accomplish.
http://address/upload.html	Link works correctly in case invalid SPI flash content. Link allows upload new firmware.
http://address/cgi-bin/spi?upload.html	**This link works always!** Link shows hard coded page for uploading new firmware.
http://address/basics.asp	Link works if all act well. Link displays current firmware version and allow proceed an upgrade.

Appendix 3 - Modification and actualization

iDo units allow easy modification of their appearance and processing for current needs of application or customer. Almost all contents of web interface including HTML pages, images, styles, java scripts etc. can be changed by uploading appropriate package via web interface. Default application can be defined by same method. (Which is profitable for example while transporting greater number of identical units). Installation package is created by sending list of involved files and directories as argument into mkflashimg utility. If there is a binary image of application amongst these files it is coded and secured before sending it to the package.



After upload of package via web interface unit restarts, if there is application image amongst files, unit is automatically reprogrammed and image is deleted which creates free space in file system. Other uploaded files are available via http protocol at address of following format:

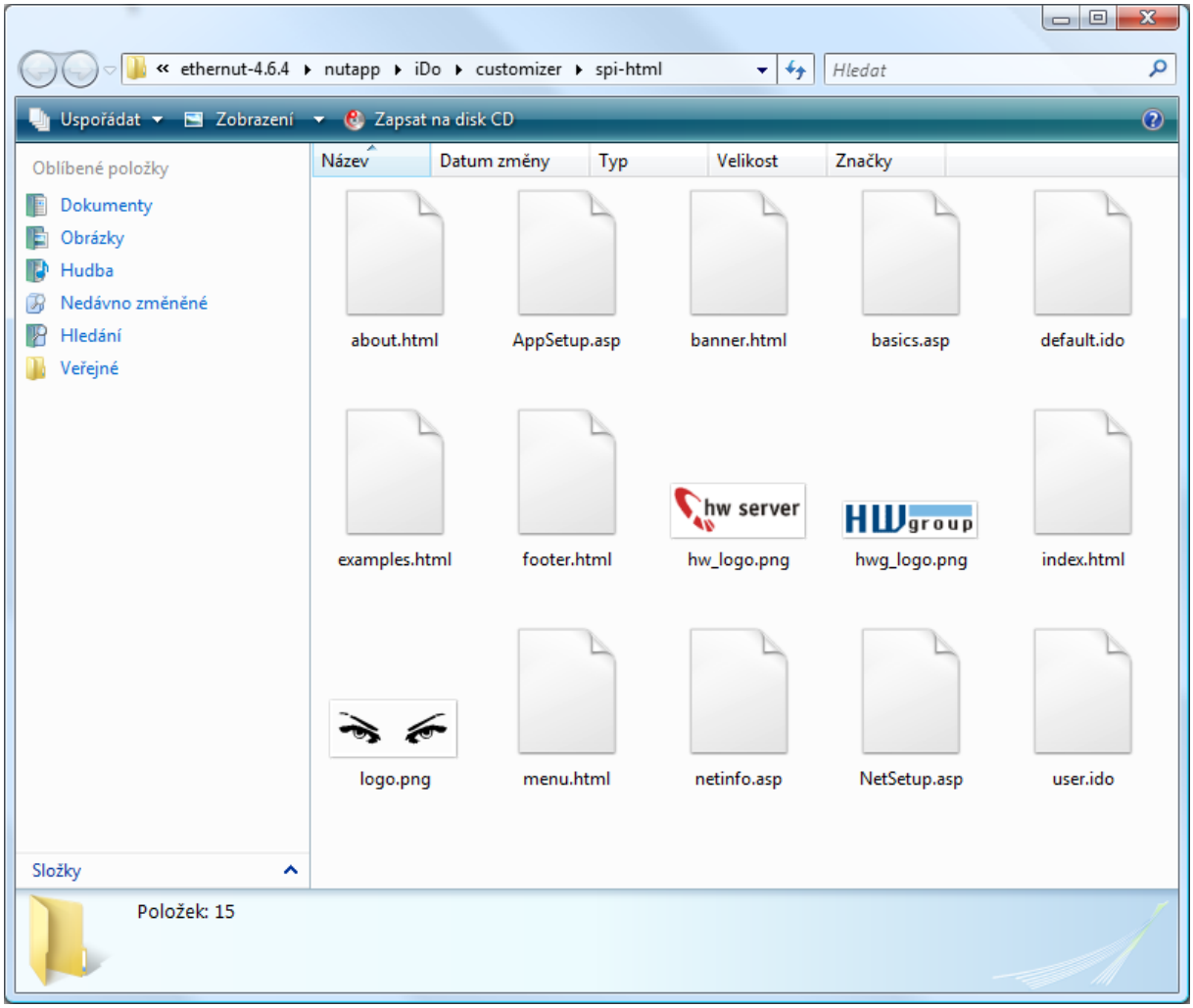
http://a.b.c.d/filename

File is transferred regarding its MIME type based on its extension. You can change whole appearance and structure of your iDo pages via this method to make it correspond with your requirements.

If there is index.html file present at file system while starting iDo it bases web root on this file. Thanks this method, you can create individual appearance of your iDo or application.

Same method may be used to access source code of currently running program too. It is saved in user.ido file. Default program (meaning the one that is loaded before user uploads anything) is accessible under default.ido file and serves for testing new units.

Default set of files:



Creation of your own pages

iDo uses standard HTML pages with several specific extensions. Default set of files may be divided into three categories:

- .html files
- .asp files
- Other files

HTML files

Standard HTML files, server does not modify them it only provides them to the user “As is”.

ASP files

If you view source code of pages in basic set you are probably going to find that .asp pages contain strange symbols in compound brackets.

Explanation is simple: Before server sends them to the user it interprets them and replaces symbols with corresponding values. In addition if .asp page is requested by GET method with parameters then these parameters are interpreted and applied on corresponding system parameters or program variables. This mechanism enables easy creation of user forms linked with application.

Parameter can be any variable, control command as well as one of following system parameters:

Parameter	Meaning
Version	Firmware version
Build	Firmware build
NutVersion	Version of current OS
Time	System time
UpTime	Time of system run
Tasks	Number of running processes
RAM	Free RAM
EthName	Name of Ethernet interface
MAC	MAC address
IP	IP address
NetMask	Network mask
Gateway	Default gateway
Syslog	IP address of Syslog server
NTP	IP address of NTP server
LogLevel	Bit mask for syslog reports
NTPTick	Interval of time check with NTP server
DIP	Values of DIP switches
DHCP	DHCP politics
Name	Name of device
Flash	Status of SPI Flash diagnostics
XMLPort	XML Port for broadcasting status information (0 = Do not send)
XMLPeriod	Period of sending regular status reports [sec]

Control commands

If expression in brackets starts with special char, is replaced according following table:

Char	Example	Meaning
#	{#file}	Simple file include. Included file is always searched in uploaded set and only in case of failure system checks UROM if file still cannot be found replacement fails. Content of the included file ISN'T interpreted.
@	{@file}	Interpreted file include. Included file is always searched in uploaded set and only in case of failure system checks UROM if file still cannot be found replacement fails. Content of the included file IS interpreted.
?	{?variable}	Conditional output based on variable value. Output is disabled if variable == 0, enabled if variable != 0.
!	{!}	Switch output status. Enable output if disabled and vice versa.
.	{.}	Enable output.
*	{*variable}	Set mask value according variable.
&	{&variable}	Enable output if (mask & variable) != 0.
=	{=variable}	Enable output if (mask == variable).
%	{%ascii}	Replace expression with character corresponding to given ASCII value

Other files

Other files are transferred without changes correct MIME type is based on extension of file name. Unrecognized types are transferred as "text/plain; charset=iso-8859-1".

System recognizes following types of files:

Ending	MIME type	Usage
".txt"	"text/plain"	Casual text files(readme atd.)
".html"	"text/html"	Static Web pages
".shtml"	"text/html"	Secured static web pages
".asp"	"text/html"	Active server pages
".htm"	"text/html"	Static Web pages
".gif"	"image/gif"	Images
".jpg"	"image/jpeg"	Images
".png"	"image/png"	Images
".pdf"	"application/pdf"	Documents and documentation
".js"	"application/x-javascript"	Java Scripts
".jar"	"application/x-java-archive"	Java applets
".css"	"text/css"	Cascade styles
".xml"	"text/xml"	XML data a documents

Creation of default applications

There are files with „.ido“ extension amongst other files in installation pack. These files are used for saving of source codes of application scripts (programs). From application creation point of view these two are the most important:

File	Description
„default.ido“	It is able to contain any program which is uploaded and processed until user saves his/her own.
„user.ido“	This file is used for saving user-made program. Size of this empty file sets maximal size of user program.

Processing of contents including firmware:

```
Příkazový řádek
C:\ethernut-4.6.4\nutapp\iDo>mkflashimg .\spi-html homeart.bin
mkflashimg 2.1
Node Descriptor Version: 52
Creating output file package.img ...
.\spi-html is a directory, diving ...
File about.html ...
Processing .\spi-html\about.html (578B) ...
Packing as about.html ...
Reading ... Checking ... Writting ...
File AppSetup.html ...
Processing .\spi-html\AppSetup.html (737B) ...
Packing as AppSetup.html ...
Reading ... Checking ... Writting ...
File banner.html ...
Processing .\spi-html\banner.html (245B) ...
Packing as banner.html ...
Reading ... Checking ... Writting ...
File basics.html ...
Processing .\spi-html\basics.html (973B) ...
Packing as basics.html ...
Reading ... Checking ... Writting ...
File default.ido ...
Processing .\spi-html\default.ido (61B) ...
Packing as default.ido ...
Reading ... Checking ... Writting ...
File examples.html ...
Processing .\spi-html\examples.html (1466B) ...
Packing as examples.html ...
Reading ... Checking ... Writting ...
File footer.html ...
Processing .\spi-html\footer.html (97B) ...
Packing as footer.html ...
Reading ... Checking ... Writting ...
File hwg_logo.png ...
Processing .\spi-html\hwg_logo.png (512B) ...
Packing as hwg_logo.png ...
Reading ... Checking ... Writting ...
File hw_logo.png ...
Processing .\spi-html\hw_logo.png (789B) ...
Packing as hw_logo.png ...
Reading ... Checking ... Writting ...
File logo.png ...
Processing .\spi-html\logo.png (1119B) ...
Packing as logo.png ...
Reading ... Checking ... Writting ...
File menu.html ...
Processing .\spi-html\menu.html (734B) ...
Packing as menu.html ...
Reading ... Checking ... Writting ...
File netinfo.html ...
Processing .\spi-html\netinfo.html (420B) ...
Packing as netinfo.html ...
Reading ... Checking ... Writting ...
File NetSetup.html ...
Processing .\spi-html\NetSetup.html (1141B) ...
Packing as NetSetup.html ...
Reading ... Checking ... Writting ...
File user.ido ...
Processing .\spi-html\user.ido (4000B) ...
Packing as user.ido ...
Reading ... Checking ... Writting ...
Processing homeart.bin (123924B) ...
Firmware size: 123924B (Remains: 3052B)
Reading ...
Session key: 0x35a968a7, Encrypting ...
Checking ... Writting ...
Done.

C:\ethernut-4.6.4\nutapp\iDo>
```

If you manage to upload something you did not wanted to and you have no way to access original upload form, nothing is lost you will find it at http://a.b.c.d/upload.thml.

Appendix 4 – HTTP interface

iDo offers its http interface not for web presentation only but for accessing many useful information too. You can gain access to following data:

- XML file with all variables data
- Source code of stored program
- Definition and settings of 1W sensors
- File with stored values of user variables
- Log file
- Generally any file stored in SPI or CPU flash memory
- Dynamic documents created via ASP and XML templates from SPI flash memory
- Etc.

Common URL syntax to obtain file is:

http://iDoAddress/filename

Files with extension ASP and XML are interpreted before processing according to rules for ASP files described in appendix 3.

Some useful filenames:

File name	Format	Contain
user.ido	text	User defined program source code.
default.ido	text	Default (factory) program source code.
vars.var	text	Stored user defined variables.
sensors.sns	binary	Sensors description and settings.
ido.log	text, csv	Datalogger data (some version).
message.xml	text	XML file with current state of iDO device.

Appendix 5 – Command line

Time after time you cannot control iDo device personally or you prefer direct communication with program (running on PC) without iDo programming, typically in case of setting some variable or remote log of measured values.

For all situations mentioned above you can use advantage of many programs which support http protocol directly from command line. I personally suggest nice and well documented freeware program wget.

It allows you communicate directly with your iDo "without human touch".

Some example:

Command	Effect
wget –O ido.xml http://adresa/message.xml	Download status XML into file ido.xml
wget –O program.ido http://adresa/user.ido	Download source code of your program into program.ido file

Remote variable setting

What about iDo remote control possibility? Yes, it is possible quite simply and smartly. ASP processor inside iDo process all requests about ASP and XML pages, fully interprets whole request including "form data" inside GET and POST methods prior page creation.

If data make sense are using beside its relation to page or file.

Length of data inside GET method is limited to 255 characters. Length of data inside POST method is limited available RAM memory only.

Example:

wget -O ido.xml http://192.168.0.178/message.xml?huh=42

It creates or sets variable "huh" to value 42 and then store current state of all variables into file ido.xml.

Operation progress:

```
C:\Users>wget -O ido.xml "http://192.168.0.178/message.xml?huh=42"
--08:13:44--  http://192.168.0.178/message.xml?huh=42
           => `ido.xml'
Connecting to 192.168.0.178:80... connected!
HTTP request sent, awaiting response... 200 Ok
Length: unspecified [text/xml]

    0K .                                                      @  64.94 KB/s

08:13:44 (62.85 KB/s) - `ido.xml' saved [1995]
```

New file named "ido.xml" will contain following data:

```xml
<?xml version="1.0" encoding="UTF-8"?>
- <device>
    <app>iDo</app>
    <version>5.09 Net</version>
    <build>May 4 2009 12:33:40</build>
    <name>ido</name>
    <event>Request</event>
- <variables>
      <huh>42</huh>
      <dip.2>1</dip.2>
      <dip.1>1</dip.1>
      <in.4>0</in.4>
      <in.3>0</in.3>
      <in.2>0</in.2>
      <in.1>0</in.1>
      <out.2>0</out.2>
      <out.1>0</out.1>
      <prog.cycle>(null)</prog.cycle>
      <sys.UpTime>55890</sys.UpTime>
      <sys.time>1241510500</sys.time>
      <led.2>2</led.2>
      <led.1>2</led.1>
      <prog.period>1000</prog.period>
      <prog.len>1</prog.len>
      <prog.steps>0</prog.steps>
    </variables>
  </device>
```


As you can see, XML file contains variable you set including its value.

More variables at once

Of course, you can set more than one variable just using one command. In this case, you must separated each couple “variable=value” with ampersand (&).

Example:

wget -O ido.xml „http://192.168.0.178/message.xml?huh=42&out.1=1”

It not only set variable huh but simultaneously switch on output 1.

ATTENTION!

If you indeed set outputs directly your program MUST run, otherwise variables will be set but physical output NOT.

We suggest enclose longer URL into quotation marks.

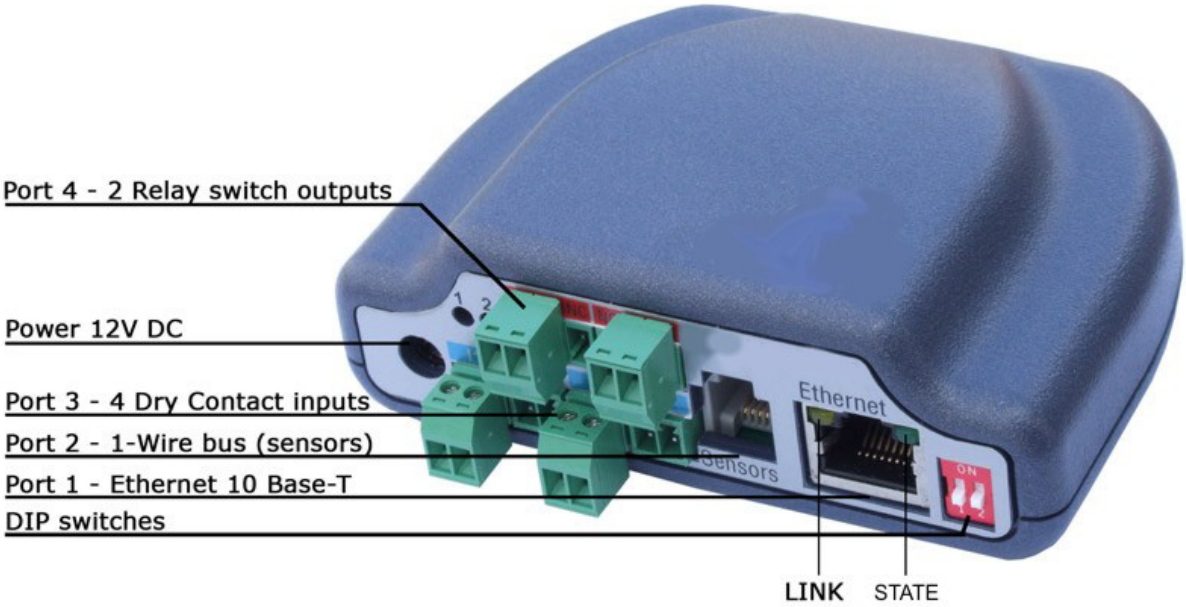
How to get wget

Probably direct from source, at:

http://www.gnu.org/software/wget/

There are not only Windows distribution package (Vista compatible) but Linux too and more documentations with examples.

Appendix 6 - Technical data



- **Port 1:** Ethernet RJ 45 - 10BASE-T / 10 Mbit/s
- **Port 2:** SENSORS - 1-Wire bus (RJ12)
- **Port 3:** 4 Dry Contact input for contacts connection
- **Port 4:** 2 switching outputs of inner relays
- **Power:** 12V DC, max. 250mA

Table values

Port 1 - Ethernet port	
Interface	RJ45 (10BASE-T) – 10 Mbit or 10/100 Mbit network compatible
Supported protocols	IP: ARP, TCP/IP, HTTP, UDP/IP, SNTP, Syslog, SMTP, XML
Port 2 - 1-Wire sensor bus	
Connector	RJ12
Sensors / distance	8 sensors, up to distance of 10 m
Port 3 - Dry contact inputs	
4 Contact inputs	For direct contact connection (dry contact).
Max distance	Up to 30 m
Input current	Max. 20mA
Port 4 - Relay switch outputs	
2 Digital outputs	Switching relay
Isolation	Galvanic separated up to 50V DC
Type of contacts	Switch NO, NC
Max. charge	1A at 24V DC; 0.3A at 50V DC; 0.5A at 50V AC
LED Status	
STATUS	**Green** – Software controlled (LED variable)
LINK & Activity	**Yellow** - Ethernet connection and activity
DIP SWITCH	
DIP1	Readable via network (http, XML), usable in program script
DIP2	Readable via network (http, XML), usable in program script
Physical parameters	
Charging	12 - 15V / 250 mA DC coaxial charge connector (barrel), Ground for shading
Size	35 x 101.6 x 76.2 [mm] (H x W x D)
Weight	150 g