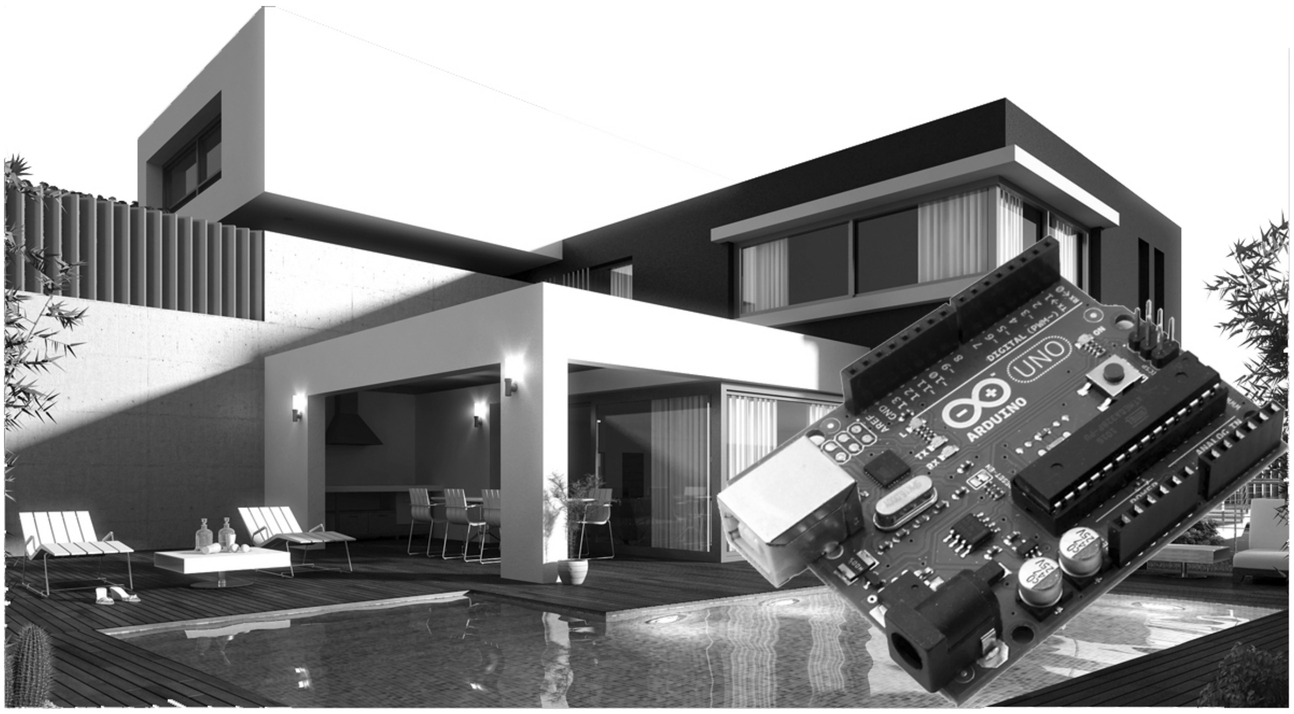
DomoS User Manual

An open-loop control system arduino-based for home automation



How to use and starting guide

Index

1 – Introduction

Congratulation for having installed DomoS on your arduino.

DomoS is an open-loop control system for home automation created by a student for his graduation exam and distributed under the term of the GPLv3.

DomoS allows you to control some peripherals such as lights, appliances and anything else for which you can create an actuator.

If you want an easy to use system, this is not the right place, in fact to make DomoS work you need some DIY skills
For example for turning a light on and make it stay on, you'll need to build an addressing module and a logic circuit (we will see it on the third chapter).

Now let's start.

2 – The first start

On the first start DomoS will ask you to enter some data such as the number of address pins to use, which pin to use and where to save data (for the version on which this manual is based, it can only be saved into EEPROM).

Answer all the questions and start to use DomoS.

Only one thing, the first address pin you set is the MSB for the addressing, you will understand the reason of that in the next chapter.

3 – The base commands

There are six base commands which you can use.

To using them you need to write the command on the serial monitor of the arduino IDE.

The most important commands are 'create' and 'turn'.

'create' lets you create a peripheral.
You can set a custom name and/or address.
Syntax: create [name test] [as 42/b00101010]
'name' and 'as' are optional, if you don't set that DomoS will pick for you an address and a name.
The maximum length for a name is 9 characters.
The number for the 'as' parameter depends on how much address pin you set on the first start (if you chose 3 address pin, as can go from 1 to 2^3-1), the number can also be set as binary using the keychar b before the binary string (remember not to put a space).
The binary representation will be the address for that peripheral.

'turn' allows you to activate a peripheral with a logic signal or an analogue signal (the arduino uno use a PWM for this)
Syntax: turn name high/low/%10/v2.
'name' is the name of the peripheral you want to activate.
The following parameter is the value you want to send.
It can be a logic value (high, low), a percentage value or a voltage value (in the version on which this manual is based,

only integer voltage value through 0 to 5 and percentage value through 0 to 100 are allowed).

There are further 4 commands, but their use is less common.

'delete' allows you to delete a peripheral.
Syntax: delete name

'list' makes a list of all the installed peripheral.
Syntax: list

'reset' deletes the content of the EEPROM and, at the next start, restart all the system.
Syntax: reset

'exit' turn off the module, in the version on which this manual is based, this command doesn't mean much.
Syntax: exit

4 – Create our first peripheral

Suppose that we want to control a digital light switch using DomoS.
That how could we do it.

First of all we need to think of a name (letting DomoS assign it for us would result only in a number) and if we want a custom address or use the automatic one.

For example, if we want to turn on the light of the kitchen, the name could be lightkitc and let DomoS chose the address for us.

Now write this on the serial monitor:

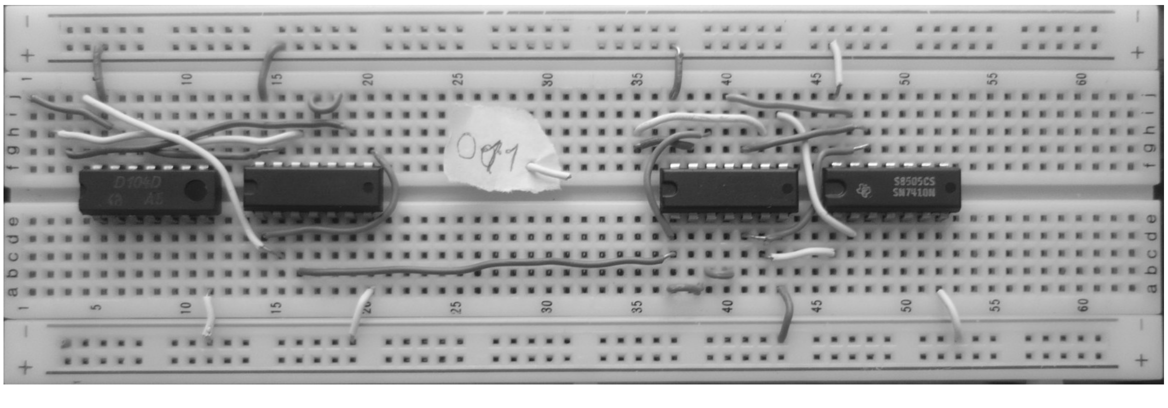
```
create name lightkitc
```

DomoS will answer, for example:

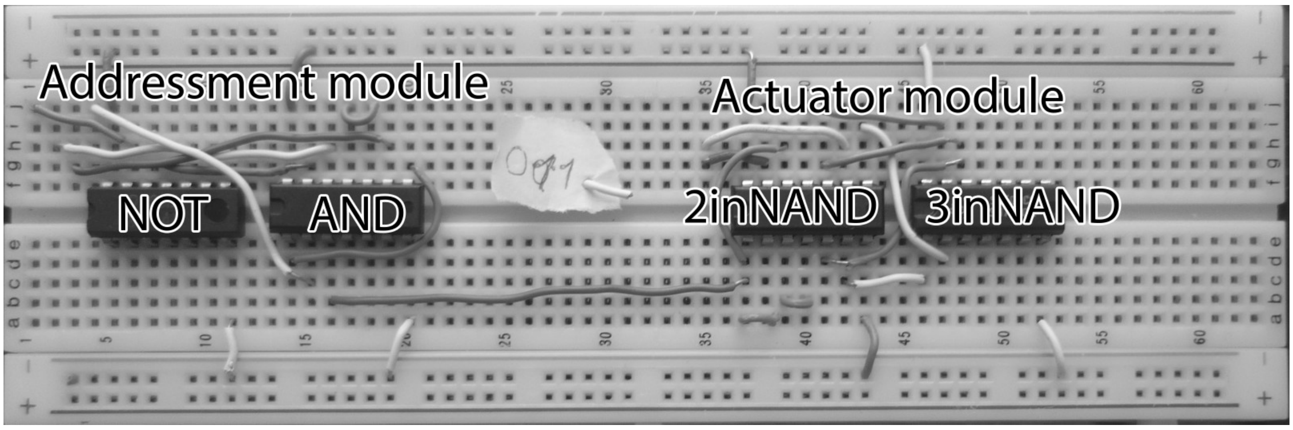
```
Peripheral lightkitc with number 3
(addressing 11) created successfully!
```

Now that the easiest part is done, let's create the actuator.

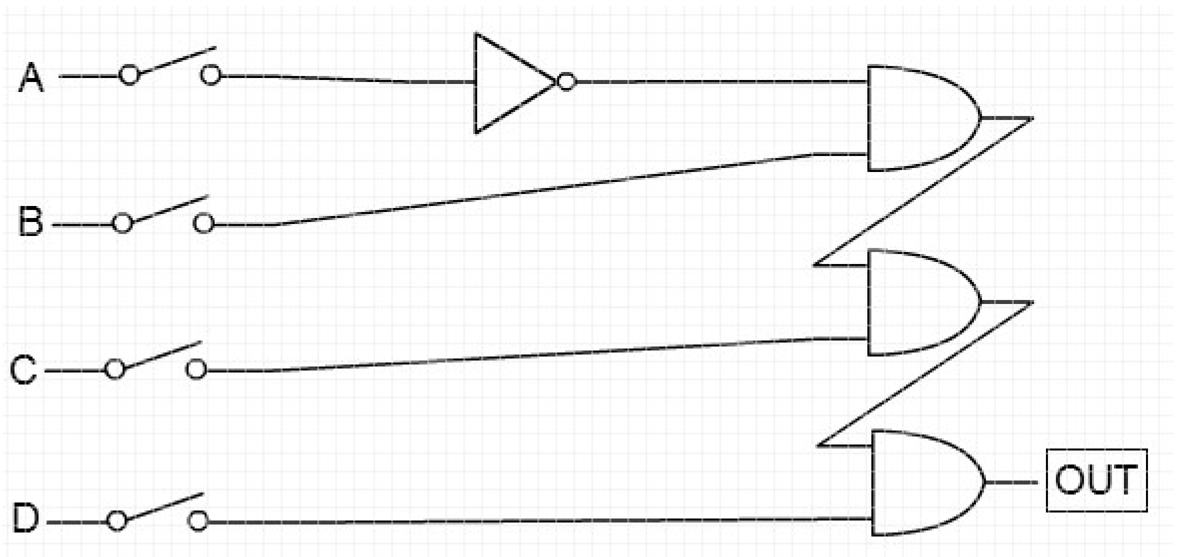
The actuator is the device that will activate our peripheral that will look like this



Let explain something



The addressing module is a kind of demux composed of some NOT gates and AND gates.
We need as many NOT gates as 0s we have in our binary representation of the address and as many AND gates as address pins we have selected plus one.
In case we selected 3 address pins, the circuit would be something like that



A, B and C are the address pins we selected earlier, D is the enabler, on the breadboard shown in the photo these are, in order, from hole f1 to hole f4, the output is on hole a16.

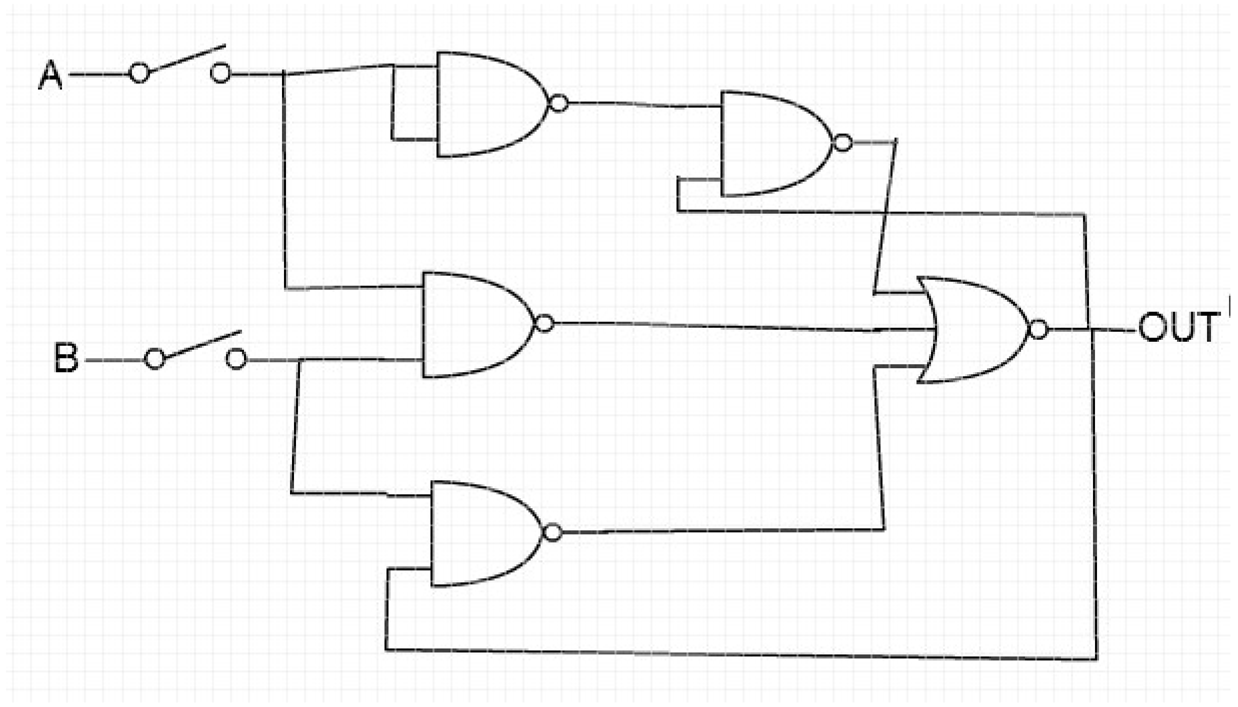
The actuator module is a bit more complex, it is a combinational circuit with feedback with this truth table

A	B	Y_{n-1}	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Simplifying the circuit we have this function:

$Y = \overline{A}Y_{n-1} + AB + BY_{n-1}$

Implementing only using NAND gates we have



It is not a very common circuit, but it works.

A is the output of the demux (on the breadboard it is in the hole a37) and B is the output of the output pin of the arduino (on hole f35), the output is on the hole f48.

If you want to test it, connect on the output of the actuator a 330ohm resistor and a led, then wire the demux and test it.

Now write on the serial monitor
```
turn lightkitc high
```
If everything went right, the led should stay on until you send
```
turn lightkitc low
```
then it should turns off.

You can build what you want, different kind of demux and actuator which accomplish what you want that DomoS do, the only limit is your electrotechnical knowledge and your fantasy.

5 – Working principles

Now we will explain what happens every time you trigger the turn command.

Suppose that we send this command

```
turn lightkitc v3
```

what happens to DomoS?

At first it will analyse your command and will see that you want to activate a peripheral, then it will search the peripheral into its database.
In this case DomoS will find the peripheral and will go further, it will analyse the value you want to send to the peripheral and will activate the output pin (be sure to connect the output pin to an RC circuit so that the PWM can be converted into an analogical value, this have a resolution of 8 bit).
Then DomoS will start to activate the addressing pin and, at last, the enabler pin.
Then it will wait for 3 seconds and it will start turning off the pin, the order is: the enabler, the addressing and the output.
Only now DomoS will accept new commands.

Now that you know how DomoS works, try making your own peripheral.