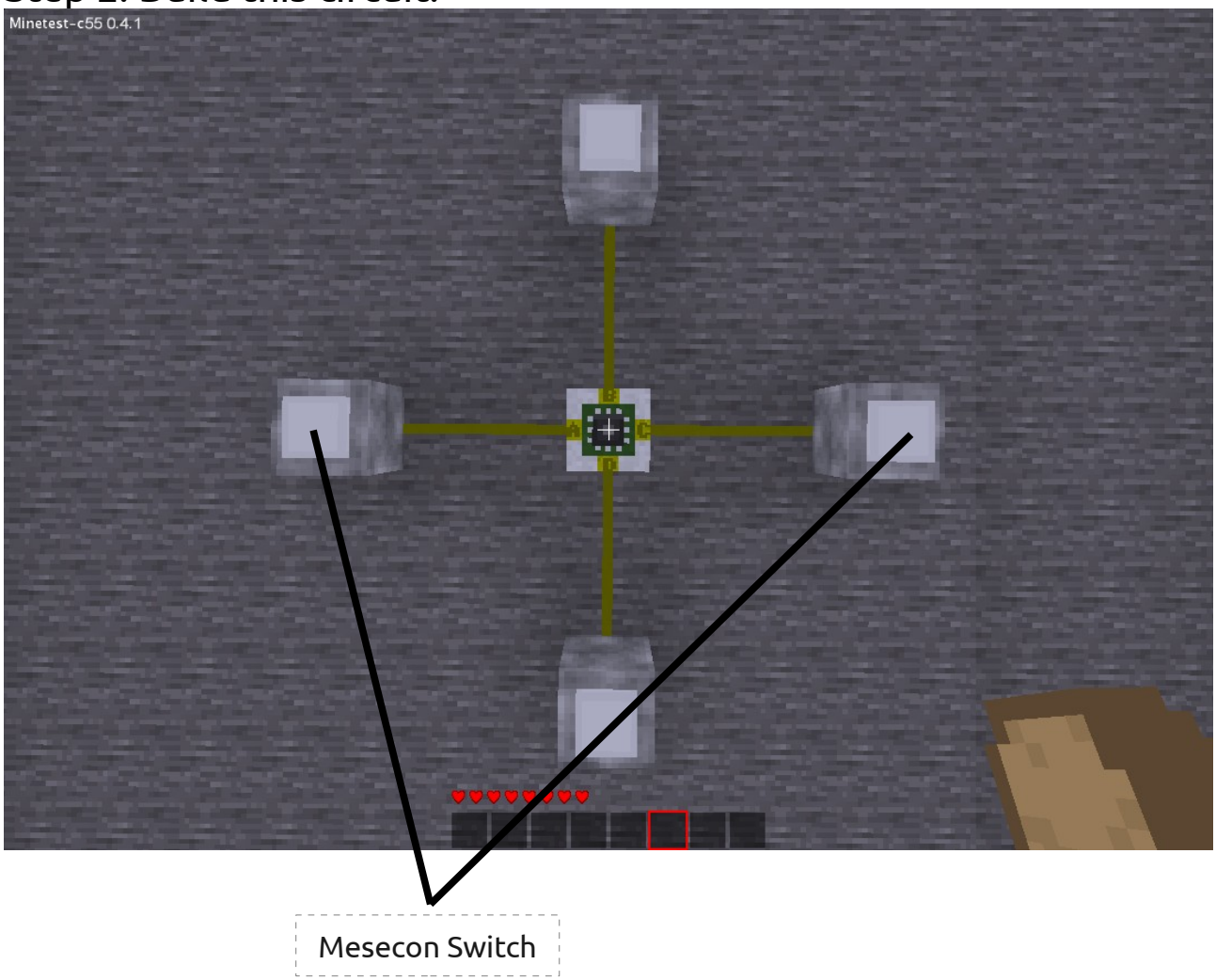
Mesecon Microcontroller coding

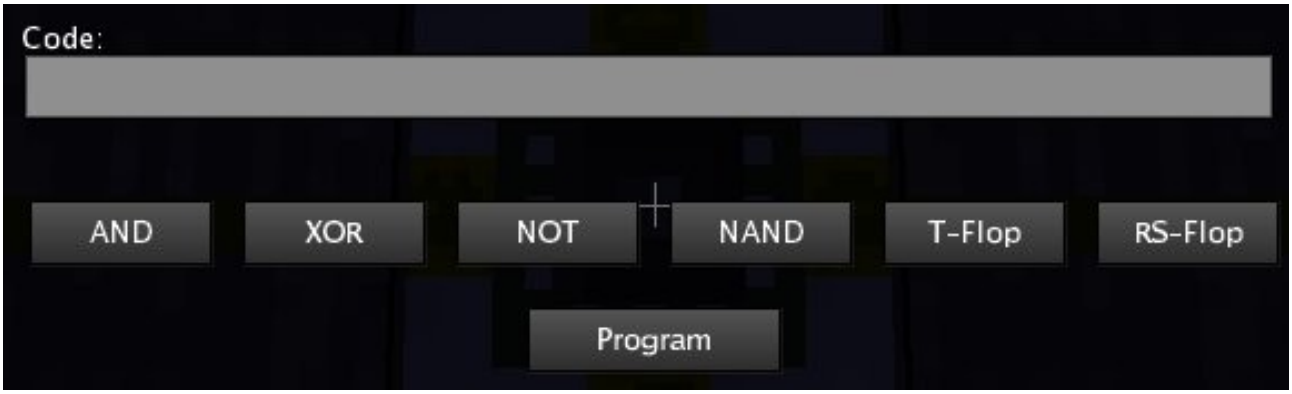
in 1 Hour (or two)

Step 1: Craft the Microcontroller

Step 2: Build this circuit:



Step 3: Right-click on the microcontroller



Step 4: Use predefined code snippets (buttons)

You can simply use predefined code snippets by selecting your logic gate from the buttons below.
The output of the ports is on under the following conditions:

- AND = Both Inputs are on
- XOR = One of the inputs is on (not both)
- NOT = Input is off
- NAND = the opposite of AND: if not both Inputs are on

There are also two FlipFlops:

- T-Flop: Toggles output if input is turned on and after that off again
- RS-Flop: Has two inputs: R and S
  Output switches to off if R is on
  Ouput switches to on if S is on

The buttons insert the source code of the gates and FlipFlops in the Code field. If you don't understand what it means, don't panic! It seems to be complicated, but it is not (maybe).
Even programmers won't understand everything:
The Mesecons Microcontroller has its own programming language, it is called ***µCScript.***

Step 5: Learn coding it!

Basic information:

1) The Microcontroller has 4 Input/Output ports: A, B, C, and D
2) You can only program it by pressing any button, not by closing the window with the esc-Key
3) Every time a button is hit, the microcontroller performs a reset, which means it turns off all the Ports
4) Don't mix lowercase and uppercase letters in your code!

5) When hovering on the microcontroller with your mouse it shows its state:
   - „Unprogrammed Microcontroller“ → It was just placed
   - „Code not valid“ → There is a bug in your code
   - „Programmed Microcontroller“ → The code works fine

Basic functions:

There are 2 basic functions that the microcontroller has:
`on(port1, port2, ...)` and `off(port1, port2, ...)`

You may start by entering
`on(B)`
into the Code field. After clicking „Program“ the lamp at Port B should be turned on.

You can also pass multiple parameters to these functions:
`on(B, D)`
will turn on both lamps.

Concatenation of functions:

You can simply put one function after the other.
For example
`on(B, D)off(B)`
will only turn on Port D, Port B will stay turned off.
Port B also won't quickly light up!
The code is first parsed, the port states are set afterwards.

Using the if-clause

If cause means that the parser/compiler checks whether a given condition applies or not. If the condition applies the functions between if and ; are executed, after that or if the condition does not apply the parser continues after ; → ; is the endif

You can also use `>` which means else.
`if(A)on(B)>off(B);`

The structure for the if clause is as follows:
`if(condition)functionA()functionB()...;functionC()`

<u>The condition</u>

A condition applies if its value is 1, it does not apply if its value is 0. Any other value creates a „not valid“ error.
The letters A, B, C and D stand for the status of the corresponding „minetest-physical“-Ports.

Example:

```
if(A) on(B);
```

If you now turn on the switch connected to Port A, the microcontroller will turn on Port B. One thing you may discover: It does not turn off Port B after having turned it on.
The solution:

```
off(B) if(A) on(B);
```

You may notice that it takes half a second until the Microcontroller activates Port B: That's because the s in „Microcontroller“ stands for speed (ok, that's stupid and not the real reason, so simply accept it).

Not

The code from „The condition“ was pretty much boring, so now comes an inverter:

```
off(B) if(!A) on(B);
```

The ! Stands for not. „If Port A is not on then turn port B on“.

And

Let's make an and gate:

```
off(B) if(A&C) on(B);
```

The & stands for and.

Or

That's pretty much useless.

```
off(B) if(A|C) on(B);
```

The | stands for or.

Xor

Xor = exclusive or. Either A or B (but not both).

```
off(B) if(A~C) on(B);
```

The ~ stands for xor.

Equals (=)

```
off(B) if(A=C) on(B);
```

Port B will be turned on if Port A has the same state as Port B. (Either both off or both on)

after()

`after(time, “code”)` is a function that can basically be used to code astable multivibrators or delayers.

Example for a delayer:
`after(1.5, "sbi(C, A)")`
Delay signal from C to A for 1.5 seconds.

Example for an astable multivibrator:
`after(2, "sbi(C, !C)")`
Toggles port C
→ This function does not work as it should if another input of the µC receives a signal

There are four things to keep in mind:

- The maximum time is 200
- The code isn't executed if the game is restarted while delaying.
- If the code in this function is invalid, "Code in after() invalid" will be displayed as soon as the µC tries to execute the code
- There can only be one after function at a time!

EEPROM:

The EEPROM is an internal memory. The mesecon Microcontroller has 255 bits of internal memory by default. By default all bits are 0, the EEPROM is reset each time the Microcontroller is programmed. It also keeps the values in the game map, so you can read them after the game has been restarted. You can acces a bit using its address.

Write bit:
`sbi(address/Port, bit/condition)`
The second parameter can either be simply 1 or 0 or a condition(just like the if condition)

The first parameter can either be an address in the Microcontroller's EEPROM or a Port of the Microcontroller (A, B, C or D)
Example:
`sbi(C, A&B)`
This is the code for an and gate (as in the code snippet).

Read bit:
Bits can be read in conditions, simply by using
`#addr`

Example:
`sbi(7, 1)if(#7)on(B);`
First this code will save 1 at the address 7, and as the value of #7 is 1 it will turn on Port B.

```
sbi(12, A)if(#12)on(B);
```
That's just a more complicated version of
```
if(A)on(B);
```

Another Example is the T-FlipFlop from the button code snippets.

Debugging:

µCScript also provides a function for debugging of your code. It is called `print()`. It outputs the values in brackets to the command line.
You can either output strings:
`print("Hello World")` (Which will output Hello World of course)
or Conditions:
`print(A&!#B)`
Strings and Conditions can be concatenated using a , :
`print("EEPROM:",#1)`

Commentary:

You may have noticed that the preset snippets also include a description. A commentary is initiated by a : and goes to the end of the whole code then:
`on(B) :This code turns on Port B`

Overheating Protection:

If the Microcontroller has to perform more than 30 operations in 0.5 seconds (60operations/second). This normally only happens when coding an endless loop.
If a microcontroller overheats it drops itself as an item.

For questions and mistakes in this tutorial (most is untested) feel free to eMail me: norrepli [at] gmail.com or ask a question in the forum.

Jeija

MeseconMicroTut 0.2