



ESP Easy firmware

2016-05-20

1 Introduction

Home automation, Internet of Things - whatever terms you heard of and are curious about, here is the fast lane to a running system without the hassle of programming yourself.

This experimental setup to have a playground to explore this fascinating world of IoT, is neither expensive nor difficult to implement.

2 Setting up the development board

To get started, you will need an *ESP-12 module* based *development board*. For the current setup I am using the NodeMCU. But chances are good that you will succeed with any other ESP-12 based board.

Flash the latest *ESP Easy firmware* by following the *instructions*.

Reset your development board by pushing the tiny button marked RST on the board.

For those of you who would like to monitor the serial output, the ESP Easy firmware will after a reset start it's boot sequence. Initial information is send at 74 880 baud. Shortly after the baud rate is set to 115 200 baud.

It is important to know that after a new flashing of the firmware, the remaining flash memory will be initialized by the firmware on the first reboot in order to put in place a clean file system.

If you do not monitor the development board, simply leave it alone for 2 minutes so it can perform it's one-time initialization.

3 Basic configuration

From any laptop, tablet, cell phone or similar, scan the wireless networks available. The one we are looking for will be named *ESP_0*. Connect to it. You will be asked a password - the initial one is *configesp*. Wait till you connected successfully the development board's hotspot.

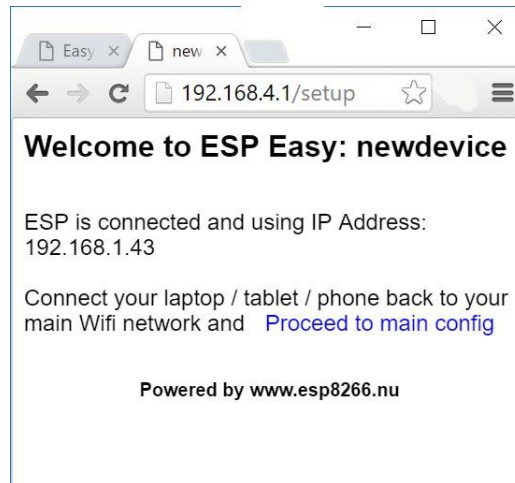
Now fire up your internet browser and enter the following address: <http://192.168.4.1/>



Setup panel

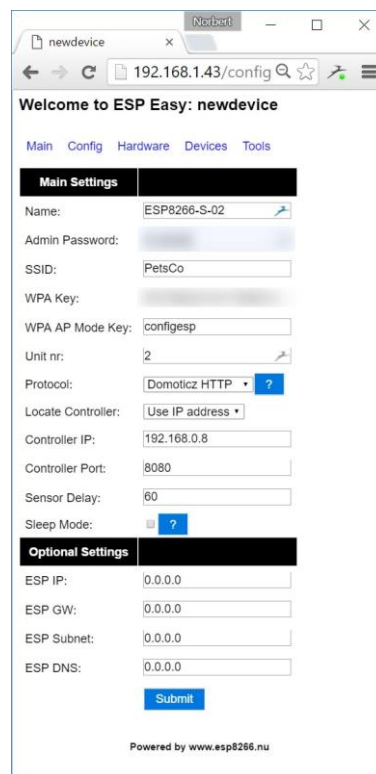


After a while the following screen will be displayed



Welcome panel

From now on the device will be connected to your local WIFI network. After clicking on the [Proceed to main page](#) link, the new IP address your development board got assigned by your internet router will be displayed in the address bar.



New device / Config panel

To make sure we are not talking different configurations afterwards, define the following input fields:

- *Name* - give the board a name of your liking



- *Admin Password* - this will be the password required for any later connection to the development board's web interface
- *Unit number* - give the board a number from 1 thru 32 - and avoid duplicates if you consider setting up more boards.

Confirm these settings by clicking the submit button.

Should you be using the *Witty Cloud (GizWits)* board, give your setup a first try - replace the *ipAddress* space holder with your development boards *<ESP IP address>*. This command will lighten up the red LED on your board.

```
http://<ESP IP address>/control?cmd=GPIO,15,1
```

This command will turn the red LED off again.

```
http://<ESP IP address>/control?cmd=GPIO,15,0
```

Another command would allow you to lighten the LED in a dimmed state.

```
http://<ESP IP address>/control?cmd=PWM,15,30
```

To dim the LED less, increase the last parameter - change the 30 to 90 for example

```
http://<ESP IP address>/control?cmd=PWM,15,90
```

Access to the blue LED (GPIO 13) or green LED (GPIO 12) is similar - simply replace the second parameter - in our case 15 - meaning GPIO 15 for the red LED.

Mixed colors can be achieved by sending several commands in sequence.

If you do not have a Witty Cloud (GizWitz) board, you have to go the extra mile and setup your breadboard and connect to a GPIO of your liking. Command line parameters must be adapted accordingly.

4 ESP Easy command line parameters

As you probably already know, your development board has GPIO pins.

4.1 Basic ON / OFF

This is what we just practiced. The commands will change the pin to high or low steady output.

- set the pin to low steady output

```
http://<ESP IP address>/control?cmd=GPIO,<pin>,0
```


- set the pin to high steady output

```
http://<ESP IP address>/control?cmd=GPIO,<pin>,1
```

4.2 PWN control

Pulse Width Modulation, or PWM, is a technique for getting analog results with digital means. Digital control is used to create a square wave, a signal switched between on and off. This on-off pattern can simulate voltages in between full on (5 Volts) and off (0



Volts) by changing the portion of the time the signal spends on versus the time that the signal spends off. The duration of "on time" is called the pulse width. To get varying analog values, you change, or modulate, that pulse width. If you repeat this on-off pattern fast enough with an LED for example, the result is as if the signal is a steady voltage between 0 and 5v controlling the brightness of the LED. 

The command we also used already:

- set the pin to a certain PWM level output

```
http://<ESP IP address>/control?cmd=PWM,<pin>,<level>
```

4.3 Short pulses

Sending a pulse signal means that the pins high state is only held for the amount of time specified.

The command is as follows:

```
http://<ESP IP address>/control?cmd=Pulse,<pin>,<state>,<duration>
```

The state may be 1 or 0. the duration is specified in milliseconds.