

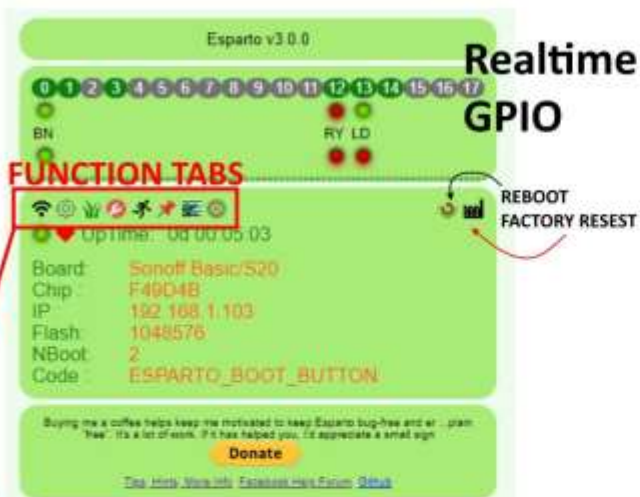


From beginner through to Master: how to develop “bombproof” ESP8266 IOT apps in minutes using the Esparto rapid application development framework

Esparto v3.0

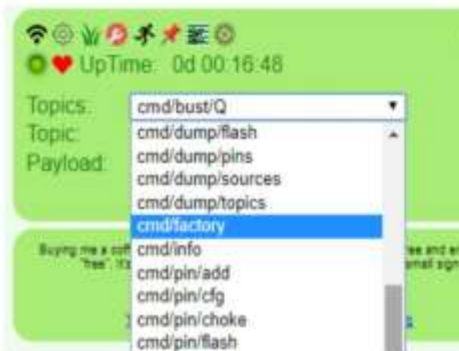
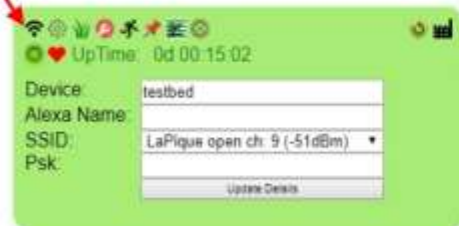
ESP8266 web UI cheat sheet

© 2019 Phil Bowles



Dynamically add GPIO

Runs on:
Wemos D1/mini/pro
NodeMCU
ESP-01/S
SONOFF Basic/S20/SV
...pretty much anything
with ESP8266 in it!



DEVICE SWITCHED BY:

- * Physical button
- * Web UI
- * Web REST command
- * Remote MQTT
- * MQTT simulator
- * Echo Voice (ALEXA)

Other Features:

- * HW functions 24/7
- * Auto fail recovery
- * No reboots
- * Easy API
- * 40+ code samples

Plus (of course!)
OTA

La Pique

Esparto v3.0.0

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

D3 D4 D1



UpTime: 0d 01:07:53

Device:	testbed
Alexa Name:	La Pique
SSID:	LaPique open ch: 9 (-60dBm) ▼
Psk:	
<button>Update Details</button>	

**GPIO
PANEL**

**TABS
PANEL
(showing
WiFi)**

Common Controls

Each icon opens a new lower tab



WiFi CPU Info Config Run Pins Log Spool

Health

UpTime: 0d 01:07:53

MQTT Heartbeat 1x per second



UI Throttled

(GPIO activity too fast to show in realtime)

Reboot

Factory Reset

WARNING!
all configuration
data will be lost



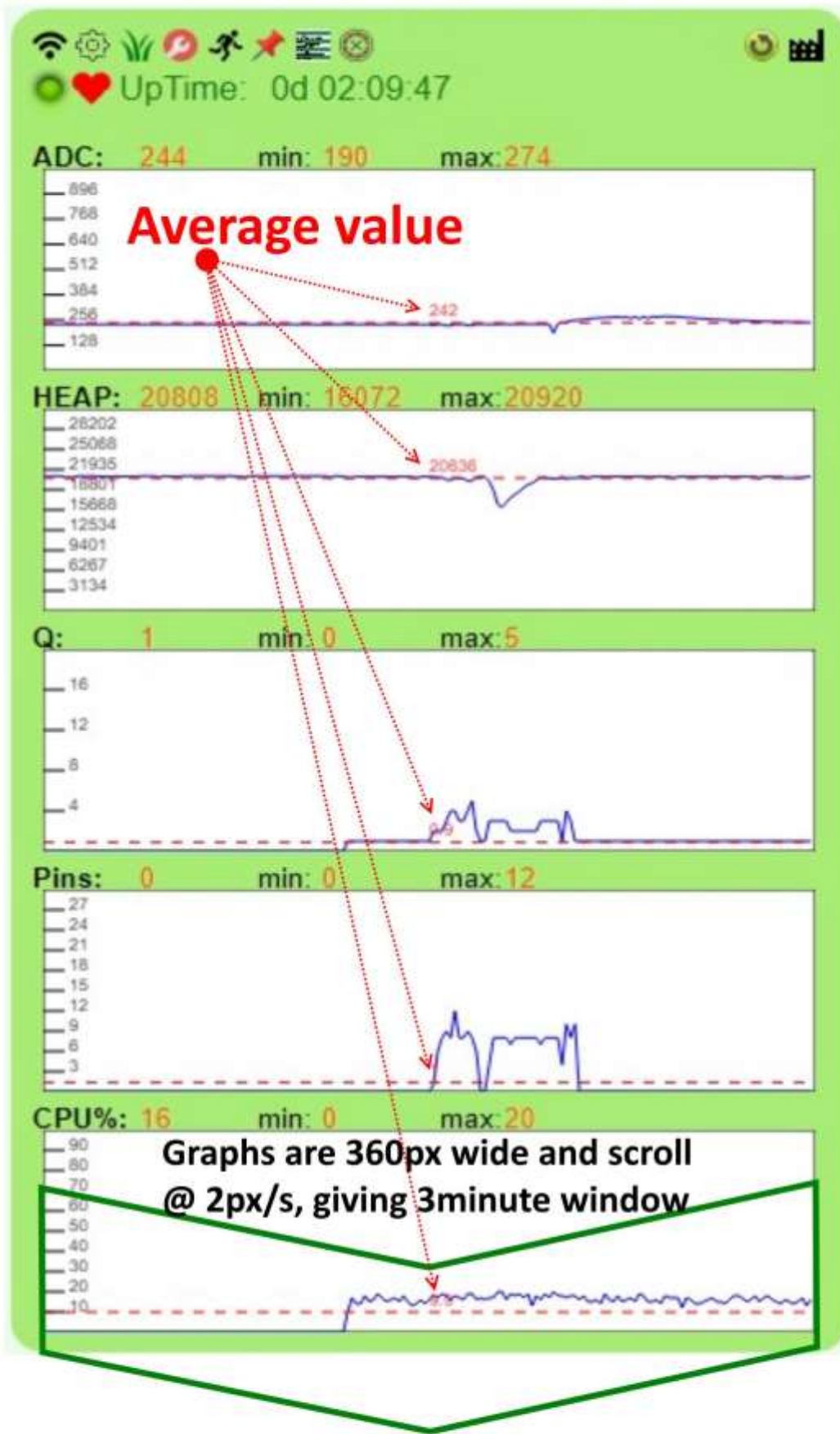
GPIO5 is Throttled

Pin is changing too fast for UI to keep up:
State is indicative and no longer real-time

New GPIO types may be added instantly
using the pins
tab

The screenshot shows the 'GPIO Pin' configuration interface. It includes a 'Pin' dropdown menu with 'click red pin #' selected, a 'Type' dropdown menu with 'select pin type' selected, an 'Action' dropdown menu with 'when pin changes...' selected, and a 'Retain' section with 'YES' and 'NO' radio buttons. A 'Create Pin' button is at the bottom.





analogRead
Value

Free Heap

Task Queue
Length

GPIO changes
per second

CPU load
percentage



The image shows a green rectangular box representing the 'Info' tab of the Esparto v3.0 Web User Interface. At the top, there is a row of icons: a Wi-Fi signal, a gear, a plant, a heart, a person running, a red arrow, a document, and a factory. Below these icons, the text 'UpTime: 0d 02:11:09' is displayed. Underneath, a list of system information is shown in orange text: 'Board: D1 Mini', 'Chip: 0BC939', 'IP: 192.168.1.114', 'Flash: 4194304', 'NBoot: 19', and 'Code: ESPARTO_BOOT_UNCONTROLLED'. A green arrow points from the 'NBoot: 19' value to the text 'Number of reboots'.

Board: D1 Mini
Chip: 0BC939
IP: 192.168.1.114
Flash: 4194304
NBoot: 19 ← Number of reboots
Code: ESPARTO_BOOT_UNCONTROLLED

```
102  
103 Esparto.reboot();  
104 // OR...  
105 Esparto.invokeCmd("cmd/reboot");  
106
```

ESPARTO_BOOT_USERCODE
ESPARTO_BOOT_UI
ESPARTO_BOOT_MQTT
ESPARTO_BOOT_BUTTON
ESPARTO_FACTORY_RESET
ESPARTO_BOOT_UNCONTROLLED



An unexpected increase in NBoots means your MCU has rebooted without it being commanded to do so.

This may be the first sign that there is a problem with your code



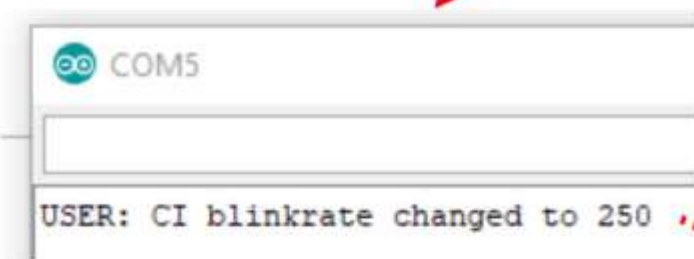
The interface shows a status bar with various icons and a heart icon indicating 'UpTime: 0d 02:12:01'. Below this is a configuration table with three rows: 'blinkrate' with value '125', 'bwf' with value 'BWF', and 'debounce' with value '10'. A red dotted arrow points from the 'blinkrate' input field to the code block on the right.

blinkrate	125
bwf	BWF
debounce	10

Changes notified to
user code automatically

```
ESPARTO_CFG_MAP defaults={  
  {"blinkrate", "125"},  
  {"bwf", "BWF"},  
  {"debounce", "10"},  
};
```

```
ESPARTO_CFG_MAP& addConfig(){  
  return defaults;  
}
```



The serial monitor shows 'COM5' at the top and a message 'USER: CI blinkrate changed to 250' below it. A red dotted arrow points from this message to the 'blinkrate' field in the configuration table below.

UI updated dynamically to reflect
any changes, no matter how caused

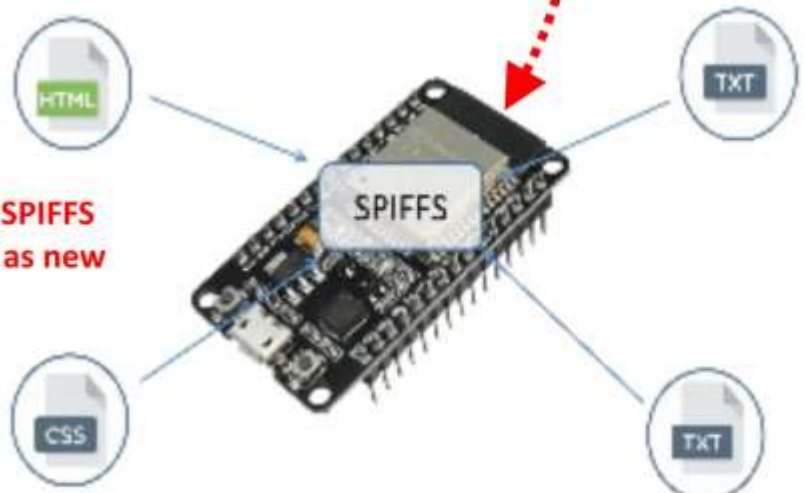


The interface is updated, showing 'UpTime: 0d 00:19:59'. The configuration table now shows 'blinkrate' as '250', 'bwf' as 'BWF', and 'debounce' as '10'. A red dotted arrow points from the 'blinkrate' field to the MQTT message on the right.

blinkrate	250
bwf	BWF
debounce	10

Dear MQTT,
variable "blinkrate"
is now 250.
Yours,
Esparto

Changes automatically saved to SPIFFS
and will re-appear on next boot as new
value



BUILT-IN

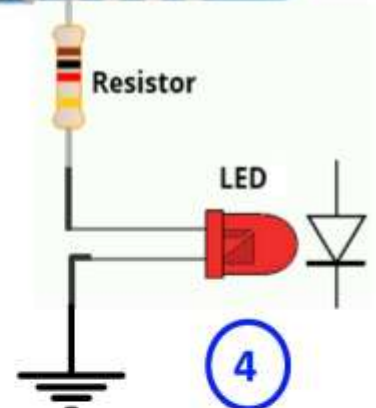
- cmd/config/get
- cmd/config/set
- cmd/factory
- cmd/info
- cmd/pin/add
- cmd/pin/cfg
- cmd/pin/choke
- cmd/pin/flash
- cmd/pin/get
- cmd/pin/kill
- cmd/pin/pattern
- cmd/pin/pwm
- cmd/pin/set
- cmd/pin/stop
- cmd/reboot
- cmd/rename
- cmd/spool

switch

User-defined

```
void onMqttConnect() {
  Esparto.subscribe("switch", mySwitchFunction);
}
```

No MQTT broker is needed: all of Esparto's cmds can run here



LED Flashes 1x per second

5 OR, just type it directly into your browser





Only RED pins can be assigned

ESPARTO_STYLE_RAW
ESPARTO_STYLE_OUTPUT
ESPARTO_STYLE_DEBOUNCED
ESPARTO_STYLE_FILTERED
ESPARTO_STYLE_LATCHING
ESPARTO_STYLE_RETRIGGERING
ESPARTO_STYLE_ENCODER
ESPARTO_STYLE_ENCODER_AUTO
ESPARTO_STYLE_REPORTING
ESPARTO_STYLE_TIMED
ESPARTO_STYLE_POLLED
ESPARTO_STYLE_DEFAULT
ESPARTO_STYLE_STD3STAGE



If you don't like programming,
this is for you!

```
<option value= "0">[when pin changes...]</option>
<option value= "1">No Action</option>
<option value= "3">Output Passthru</option>
<option value= "2">Publish Pin Value</option>
<option value= "9">Publish Var Value</option>
<option value= "4">Set Var (Pin)</option>
<option value= "5">Set Var (Param)</option>
<option value= "6">Dec Var</option>
<option value= "7">Inc Var</option>
<option value= "10">Add to Var</option>
<option value= "11">Sub from Var</option>
<option value= "8">Command</option>
<option value= "12">Flash LED</option>
<option value= "13">Flash LED PWM</option>
<option value= "14">Flash LED Pattern</option>
<option value= "15">Stop LED Flash</option>
```

Send state to an output pin

Send pin value to MQTT

Send value of variable to MQTT

Set variable to value of pin

Set variable to arbitrary value

Decrement a variable

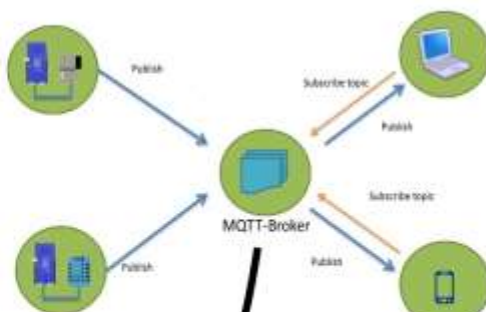
Increment a variable

Add pin value to a variable

Subtract pin value from a variable

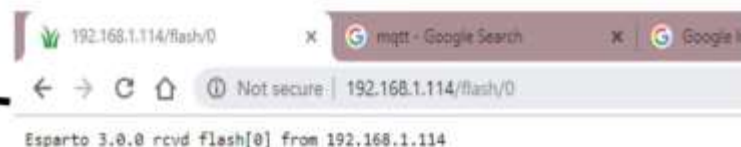
Run any command + payload





MQTT

Web UI



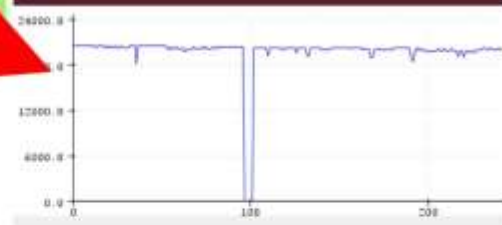
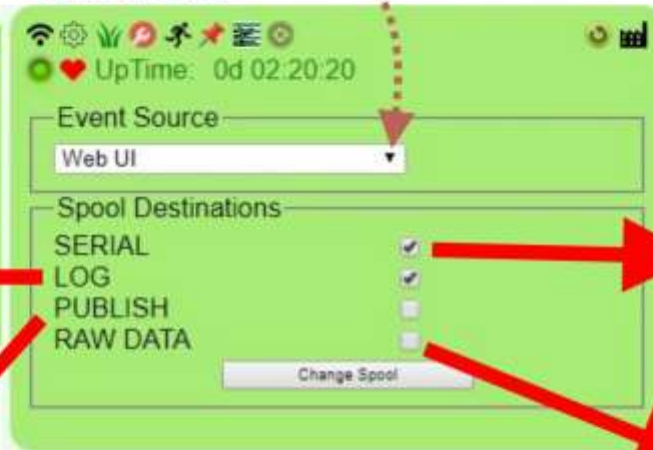
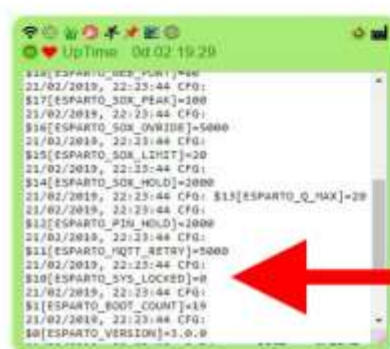
Web "REST"

User Code

Voice

Every action comes from an "event source"

GPIO Pin Activity



Esparto code runs in "layers"
Each layer handles incoming events from a different origin (or "source")

Each layer can output diagnostic information to any of a choice of destinations. Esparto calls this "spooling". You choose which type of event goes to which output

