



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

Espresso v3.0.0

**FUNCTION TABS**

BN RY LD

**Realtime GPIO**

REBOOT FACTORY RESET

Board: Sonoff Basic/S20  
Chip: F49D4B  
IP: 192.168.1.103  
Flash: 1048576  
NBoot: 2  
Code: ESPARTO\_BOOT\_BUTTON

Buying me a coffee helps keep me motivated to keep Espresso bug-free and er... plain "free". It's a lot of work. If it has helped you, I'd appreciate a small sign [Donate](#)

Tips Hints More Info Facebook Help Forum Github

UpTime: 0d 00:21:13

**GPIO Pin**

Pin	14
Type	[select pin type]
Action	[select pin type]
On Change	DEBOUNCED DEFAULT OUTPUT ENCODER ENCODER_AUTO FILTERED LATCHING OUTPUT POLLED RAW REPORTING RETRIGGERING STD 3-STAGE TIMED
Retain	

Buying me a coffee helps keep me motivated to keep Espresso bug-free and er... plain "free". It's a lot of work. If it has helped you, I'd appreciate a small sign [Donate](#)

Top Hints More Info Facebook Help Forum Github

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



## Dynamically add GPIO

UpTime: 0d 00:15:02

Device: testbed  
Alexa Name: LaPique open ch: 9 (-51dBm)  
SSID: Psk: Update Details

UpTime: 0d 00:19:47

ADC: 37 min: 28 max: 68

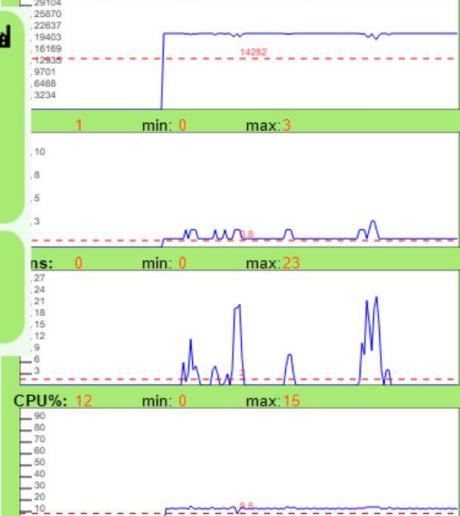
HEAP: 21336 min: 19688 max: 21448

OS: 1 min: 0 max: 3

UpTime: 0d 00:16:48

Topics: cmd/bust/Q  
Topic: cmd/dump/flash  
Payload: cmd/dump/pins  
cmd/dump/sources  
cmd/dump/topics  
cmd/factory  
cmd/info  
cmd/pin/add  
cmd/pin/cfg  
cmd/pin/choke  
cmd/pin/flash

Buying me a coffee helps keep me motivated to keep Espresso bug-free and er... plain "free". It's a lot of work. If it has helped you, I'd appreciate a small sign [Donate](#)



UpTime: 0d 00:22:09

Event Source: [select a source]  
Spool Destinations: SERIAL LOG PUBLISH RAW DATA  
Change Spool

UpTime: 0d 00:23:26

```
13/02/2019, 17:15:06 evt/USER/logs/23
13/02/2019, 17:15:02 evt/USER/spool/4
13/02/2019, 17:15:02 evt/USER/spool/22
13/02/2019, 17:14:57 evt/USER/spool/4
13/02/2019, 17:14:57 evt/USER/spool/22
```

UpTime: 0d 00:24:28

\$0	3.0.0
\$1	2
\$10	1
\$11	5000
\$12	2000
\$13	20

BareMinimum\_SONOFF\_BASIC

```
1 #include <ESPArto.h>
2 ESPArto Espresso("LaPique","", "testbed", "192.168.1.4", 1883);
3 void setupHardware() {
4   Esparto.Output(BUILTIN_LED,LOW,HIGH);
5   Esparto.DefaultOutput(RELAY,HIGH,LOW, [](int v1, int v2){Esparto.digitalWrite(BUILTIN_LED,!v1); });
6   Esparto.std3StageButton();
7 }
```

All of this from 7 lines of code!

## 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:

[Update Details](#)

**GPIO  
PANEL**

**TABS  
PANEL  
(showing  
WiFi)**

## Common Controls

Each icon opens a new lower tab



WiFi

CPU

Info

Config

Run

Pins

Log

Spool

Health

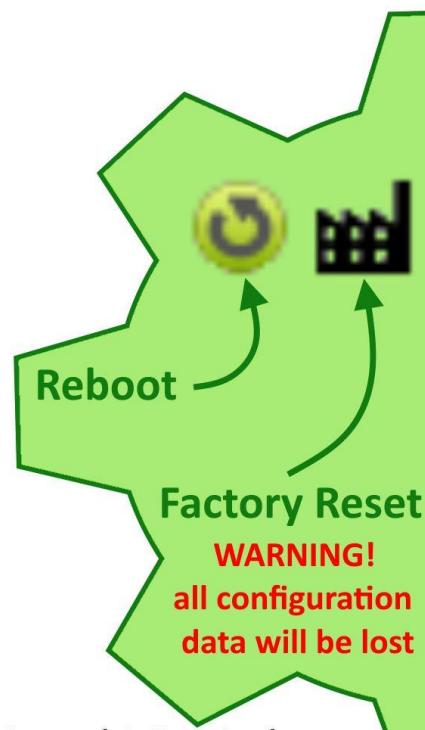
UpTime: 0d 01:07:53

MQTT Heartbeat 1x per second



**UI Throttled**

(GPIO activity too fast to show in realtime)





## GPIO5 is Throttled

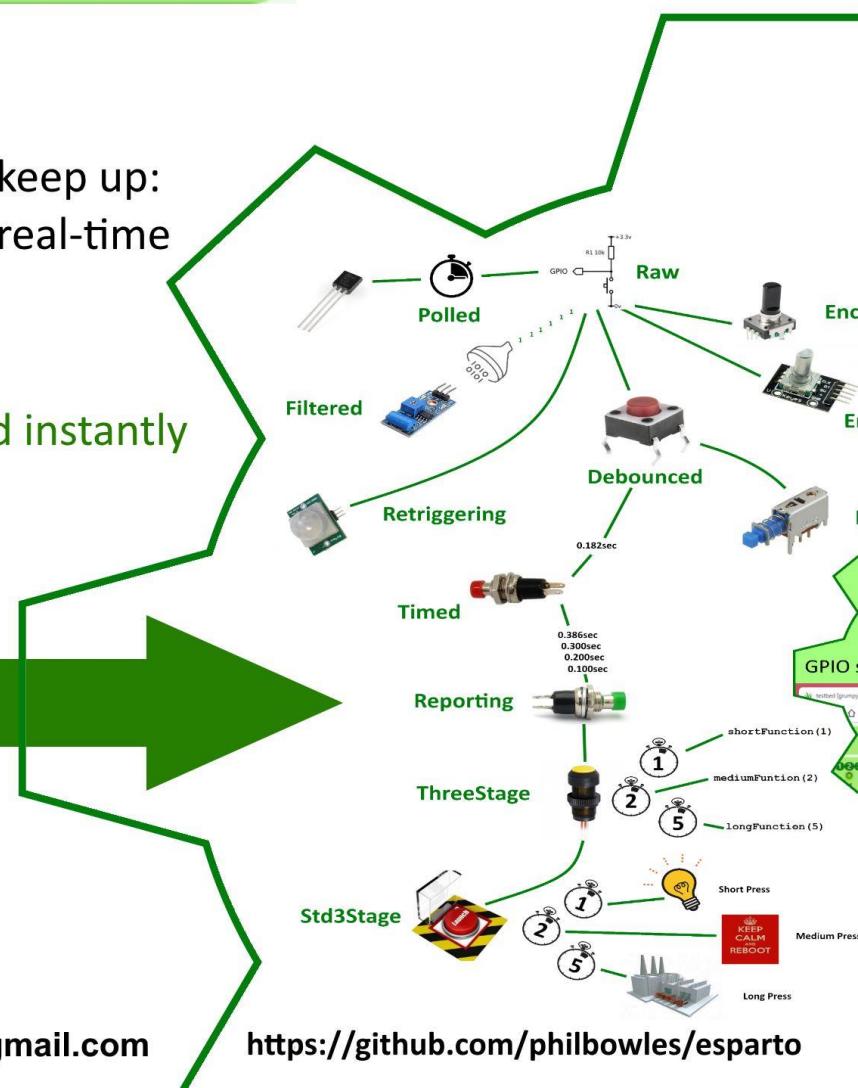
Pin is changing too fast for UI to keep up:  
State is indicative and no longer real-time

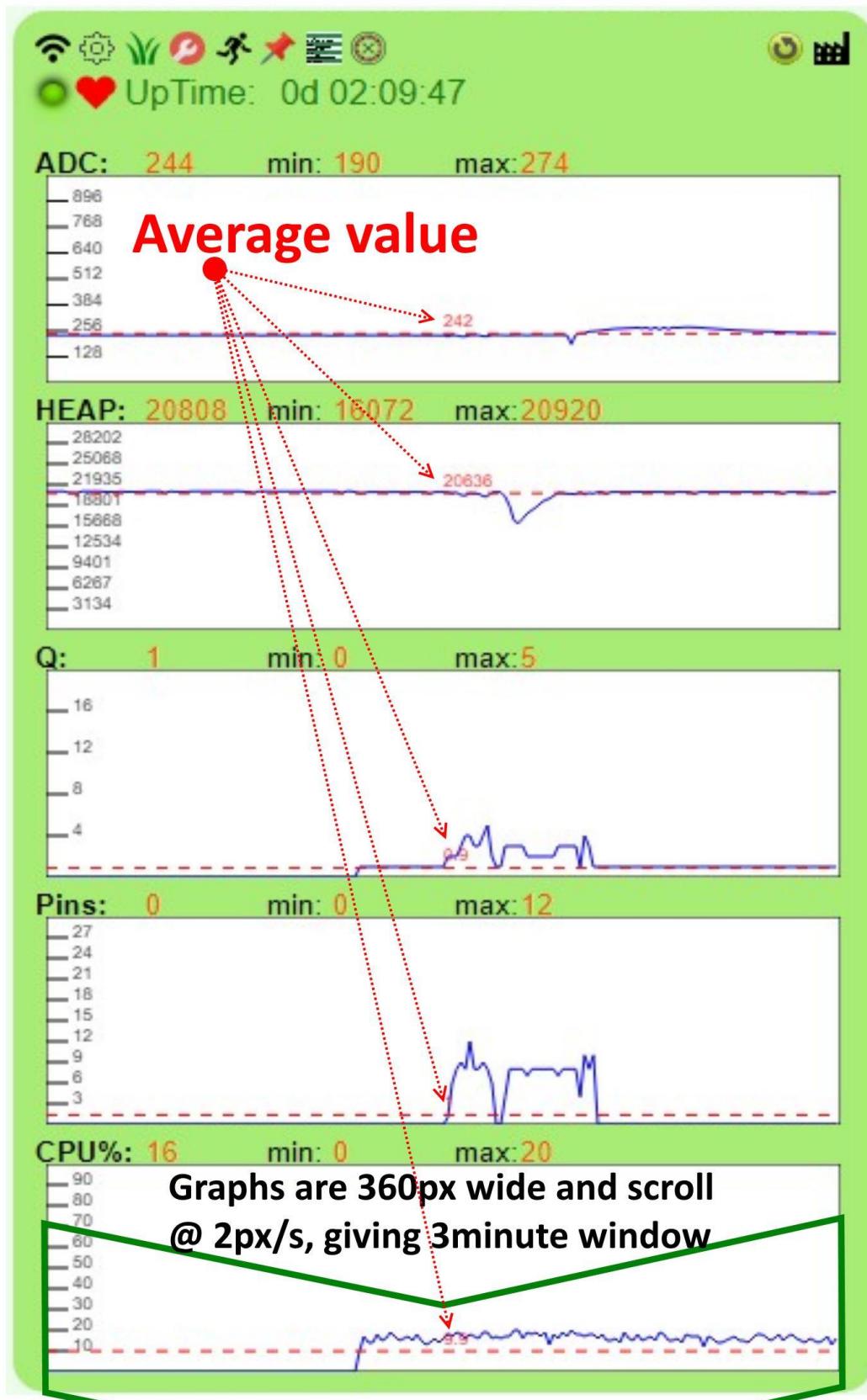
New GPIO types may be added instantly  
using  
the  
pins  
tab

GPIO Pin  
Pin   
Type

Action  
On Change

Retain  
YES  NO





analogRead  
Value

Free Heap

Task Queue  
Length

GPIO changes  
per second

CPU load  
percentage



ESPARTO\_BOOT\_USERCODE

ESPARTO\_BOOT\_UI

ESPARTO\_BOOT\_MQTT

ESPARTO\_BOOT\_BUTTON

ESPARTO\_FACTORY\_RESET

ESPARTO\_BOOT\_UNCONTROLLED

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



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

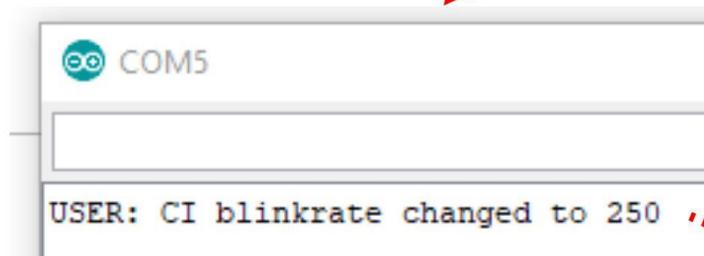
This may be the first sign that there is a problem with your code



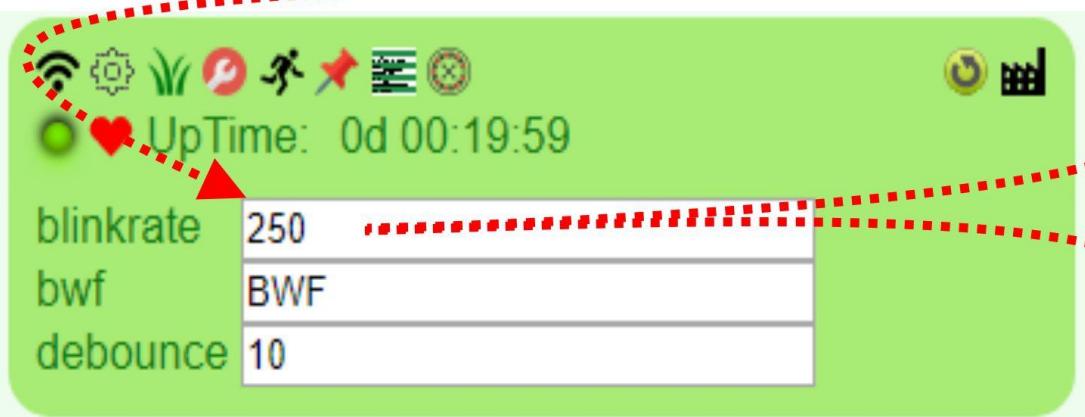
Changes notified to user code automatically

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

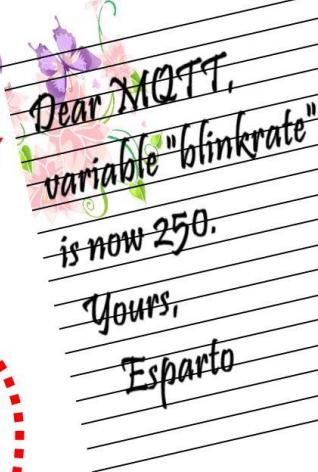
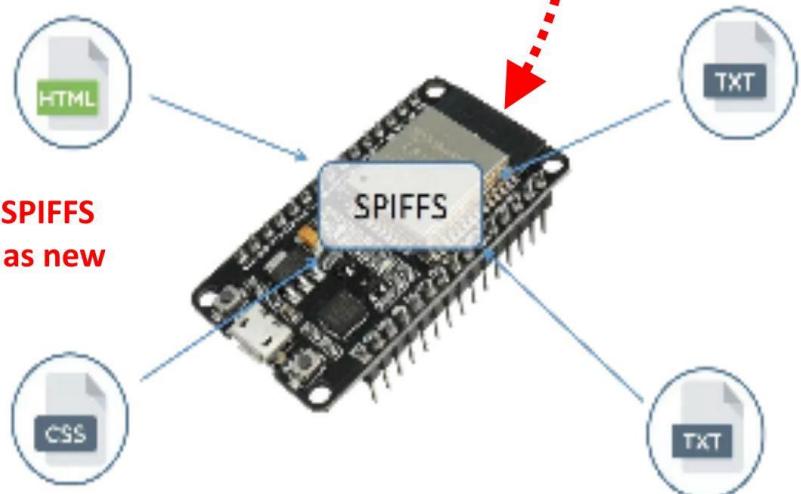
```
ESPARTO_CFG_MAP& addConfig(){  
    return defaults;  
}
```



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



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



**BUILT-IN**

Topics: cmd/bust/Q  
Topic: cmd/pin/flash/4  
Payload: 1000

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

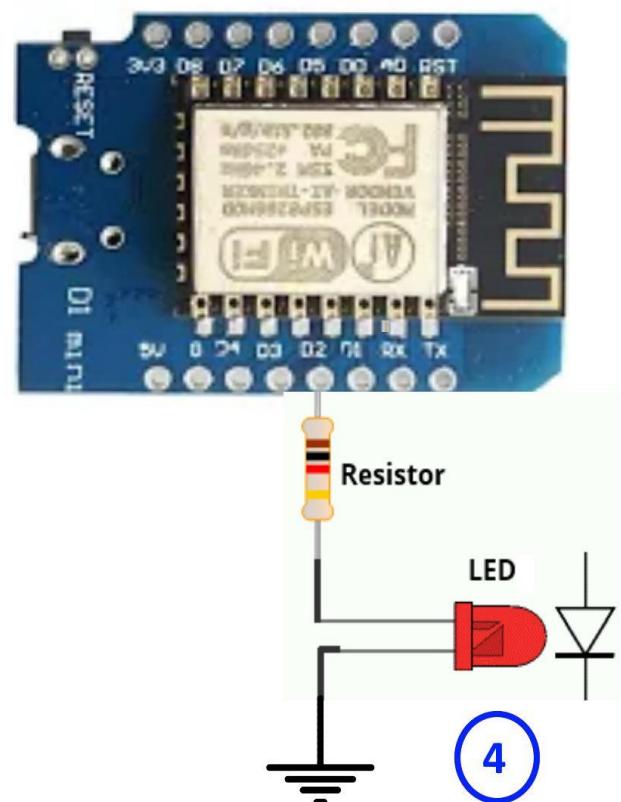
Select a topic

1

2 Add some data

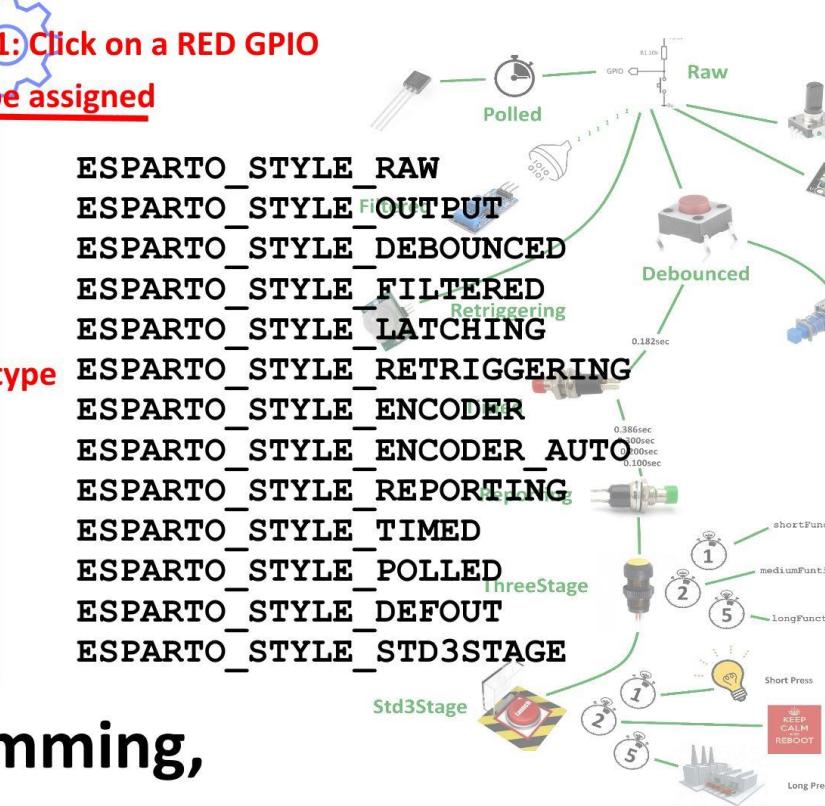
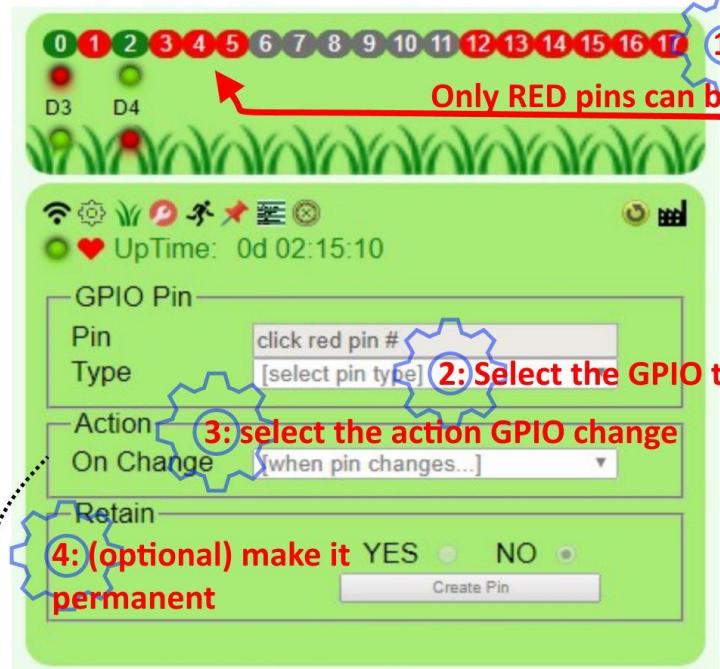
3 Click!

Simulate MQTT



5 OR, just type it directly into your browser





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

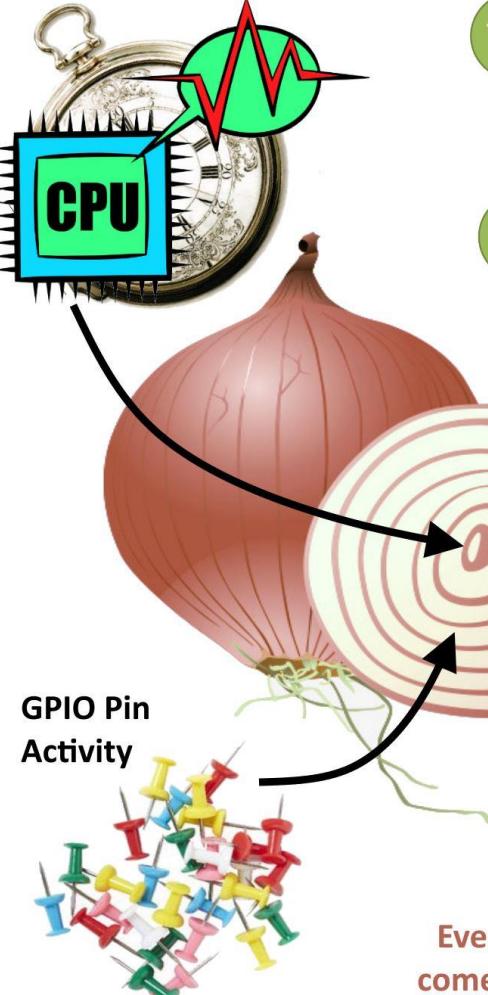
<code>&lt;option value= "0"&gt;[when pin changes...]&lt;/option&gt;</code>	<b>Send state to an output pin</b>
<code>&lt;option value= "1"&gt;No Action&lt;/option&gt;</code>	<b>Send pin value to MQTT</b>
<code>&lt;option value= "3"&gt;Output Passthru&lt;/option&gt;</code>	<b>Send value of variable to MQTT</b>
<code>&lt;option value= "2"&gt;Publish Pin Value&lt;/option&gt;</code>	<b>Set variable to value of pin</b>
<code>&lt;option value= "9"&gt;Publish Var Value&lt;/option&gt;</code>	<b>Set variable to arbitrary value</b>
<code>&lt;option value= "4"&gt;Set Var (Pin)&lt;/option&gt;</code>	<b>Decrement a variable</b>
<code>&lt;option value= "5"&gt;Set Var (Param)&lt;/option&gt;</code>	<b>Increment a variable</b>
<code>&lt;option value= "6"&gt;Dec Var&lt;/option&gt;</code>	<b>Add pin value to a variable</b>
<code>&lt;option value= "7"&gt;Inc Var&lt;/option&gt;</code>	<b>Subtract pin value from a variable</b>
<code>&lt;option value= "10"&gt;Add to Var&lt;/option&gt;</code>	<b>Run any command + payload</b>
<code>&lt;option value= "11"&gt;Sub from Var&lt;/option&gt;</code>	
<code>&lt;option value= "8"&gt;Command&lt;/option&gt;</code>	
<code>&lt;option value= "12"&gt;Flash LED&lt;/option&gt;</code>	
<code>&lt;option value= "13"&gt;Flash LED PWM&lt;/option&gt;</code>	
<code>&lt;option value= "14"&gt;Flash LED Pattern&lt;/option&gt;</code>	
<code>&lt;option value= "15"&gt;Stop LED Flash&lt;/option&gt;</code>	



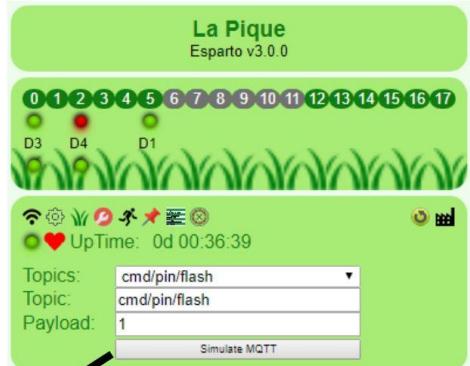
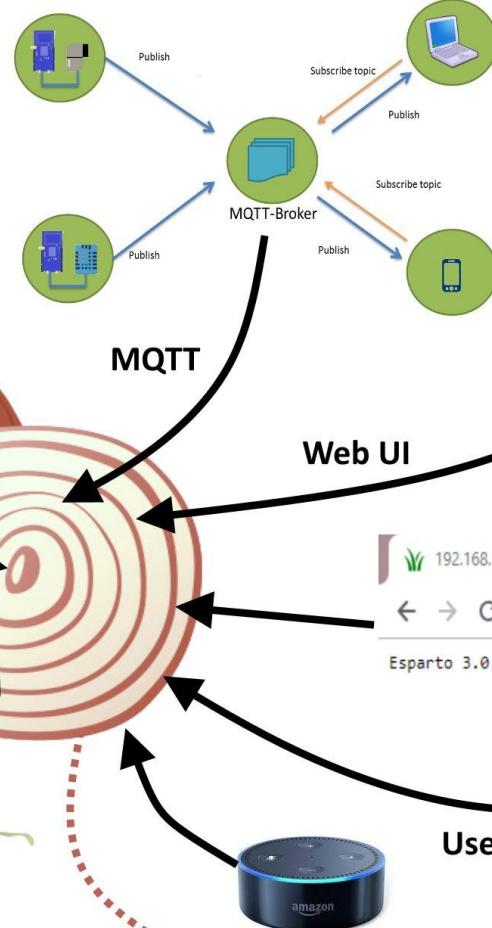
# Espresso v3.0

## Web User Interface: Spool and Log Tabs

### Scheduler & Timers



### GPIO Pin Activity



### Web "REST"

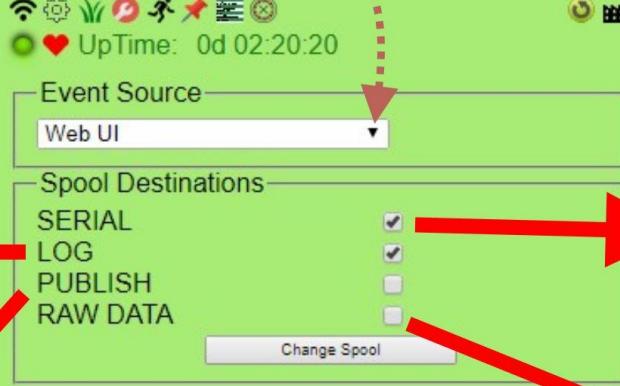


### User Code

```
MQTT_Blinky | Arduino 1.8.7
File Edit Sketch Tools Help
MQTT_Blinky
80 // const char* setAlexaDeviceName() { return "La Pique"; }
81 //
82 // What we do when Alexa calls... (button sense is reversed as GPIO is active low
83 //
84 void onAlexaCommand(bool b){ flash(b); }
85 //
86 // Default configuration parameters
87 //
88 #define ESPARTO_CFG_MAP defaults(
89   {"blinkrate","125"}, // we are going to change this later
90   {"debounce","10"}, // "bwd","BNFT"
91   {"bwf","BNFT"}
92 );
93 #endif
94 #endif
95 #endif
96 #endif
97 #endif
98 #endif
99 #endif
100 #endif
101 #endif

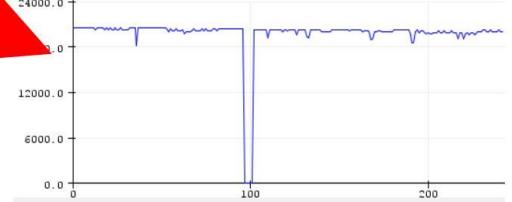
void onConfigItemChange(const char* id,const char* value){
  Serial.printf("USER: CI %s changed to %s\n",id,value);
  if(strcmp(id,"blinkrate")){
    if(Esparto.isBlinking()) flash(true);
  }
}
```

**Every action  
comes from an  
"event source"**



```
COMS
I
SS: CFG: $0[ESPARTO_VERSION]=3.0.0
SS: CFG: $1[ESPARTO_BOOT_COUNT]=37
SS: CFG: $10[ESPARTO_SYS_LOCKED]=0
SS: CFG: $11[ESPARTO_MQTT_RETRY]=5000
SS: CFG: $12[ESPARTO_PIN_HOLD]=2000
SS: CFG: $13[ESPARTO_Q_MAX]=20
SS: CFG: $14[ESPARTO_SOX_HOLD]=2000
SS: CFG: $15[ESPARTO_SOX_LOCKED]=0
SS: CFG: $16[ESPARTO_SOX_LIMIT]=20
SS: CFG: $17[ESPARTO_SOX_OVERRIDE]=5000
SS: CFG: $18[ESPARTO_WEB_PORT]=80
SS: CFG: $19[ESPARTO_MQTT_USER]=

COMS
24000.0
20000.0
16000.0
12000.0
8000.0
4000.0
0.0
0 100 200
```



**Espresso 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. Espresso calls this "spooling"  
You choose which type of event goes to which output**

