THE
DEVELOPER'S
CONFERENCE

Uma Introdução ao Mongoose OS no ESP8266

Felipe Kühne

Engenheiro de Sistemas Embarcados

fkuhne@gmail.com

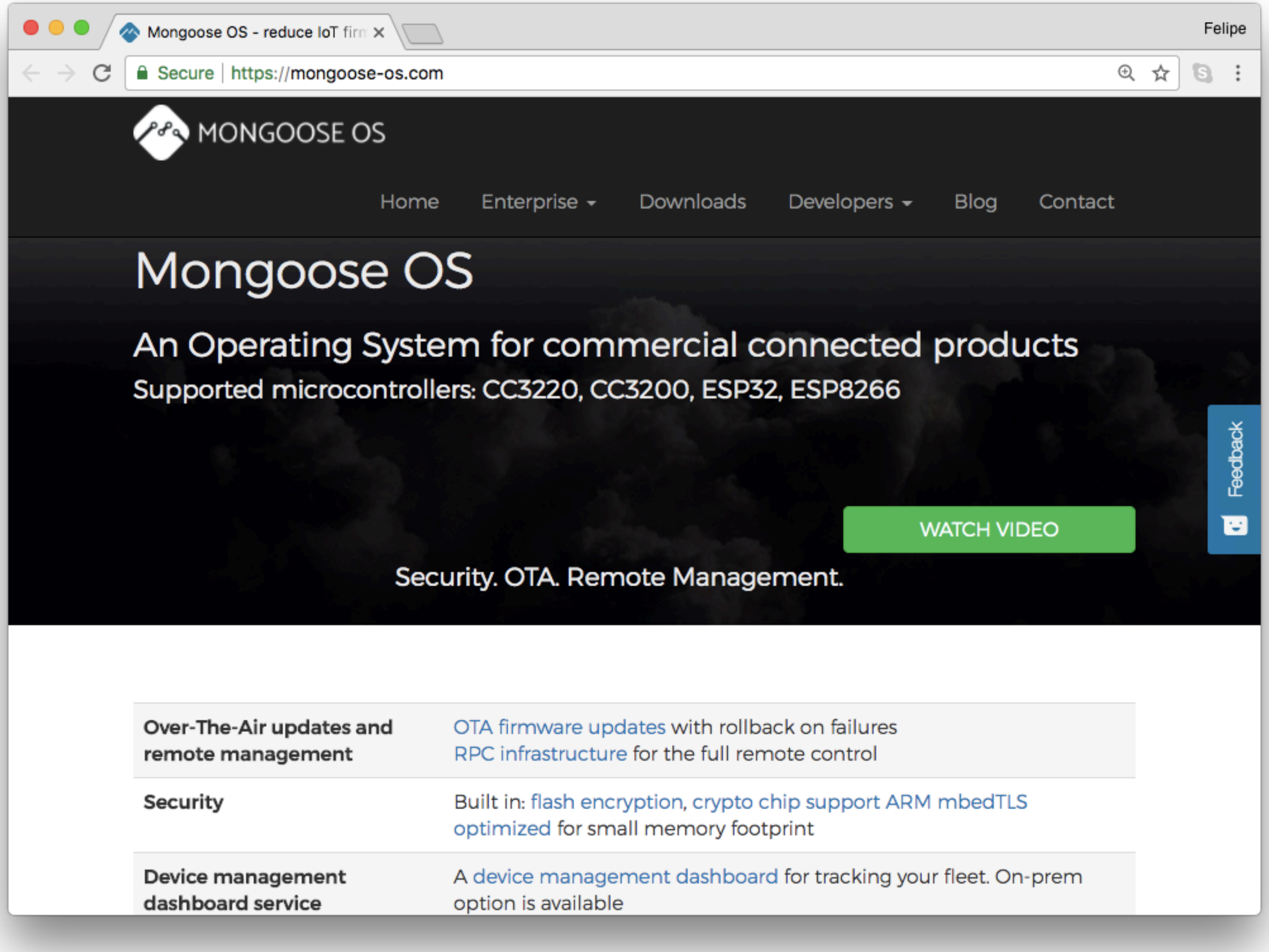
Globalcode

github.com/fkuhne/tdc17

Agenda

- Introdução
- Arquitetura
- Instalação
- Codificação e Compilação
- Exemplo: Hello World
- Exemplo: Blink com configuração remota
- Exemplo: Cliente MQTT
- Referências

Introdução



Mongoose OS

An Operating System for commercial connected products

Supported microcontrollers: CC3220, CC3200, ESP32, ESP8266

Security. OTA. Remote Management.

Over-The-Air updates and remote management	OTA firmware updates with rollback on failures RPC infrastructure for the full remote control
Security	Built in: flash encryption, crypto chip support ARM mbedTLS optimized for small memory footprint
Device management dashboard service	A device management dashboard for tracking your fleet. On-prem option is available

Introdução

- SO para devices IoT a nível comercial
- Suporta alguns hardwares: STM32, ESP, Texas, Nordic
- Compatível com AWS, Google e outros serviços de núvem
- Open source (licença comercial disponível)
- Criptografia de mensagens (TLS) e do conteúdo da flash
- Inúmeros protocolos de comunicação (HTTP, WebSockets, MQTT, CoAP, …)
- JavaScript e C/C++
- Sistema de arquivos (POSIX API)
- Over-the-air update
- Sistema de gerenciamento remoto de dispositivos
- Sistema de chamadas remotas: RPC *(remote procedure calls)*

Introdução

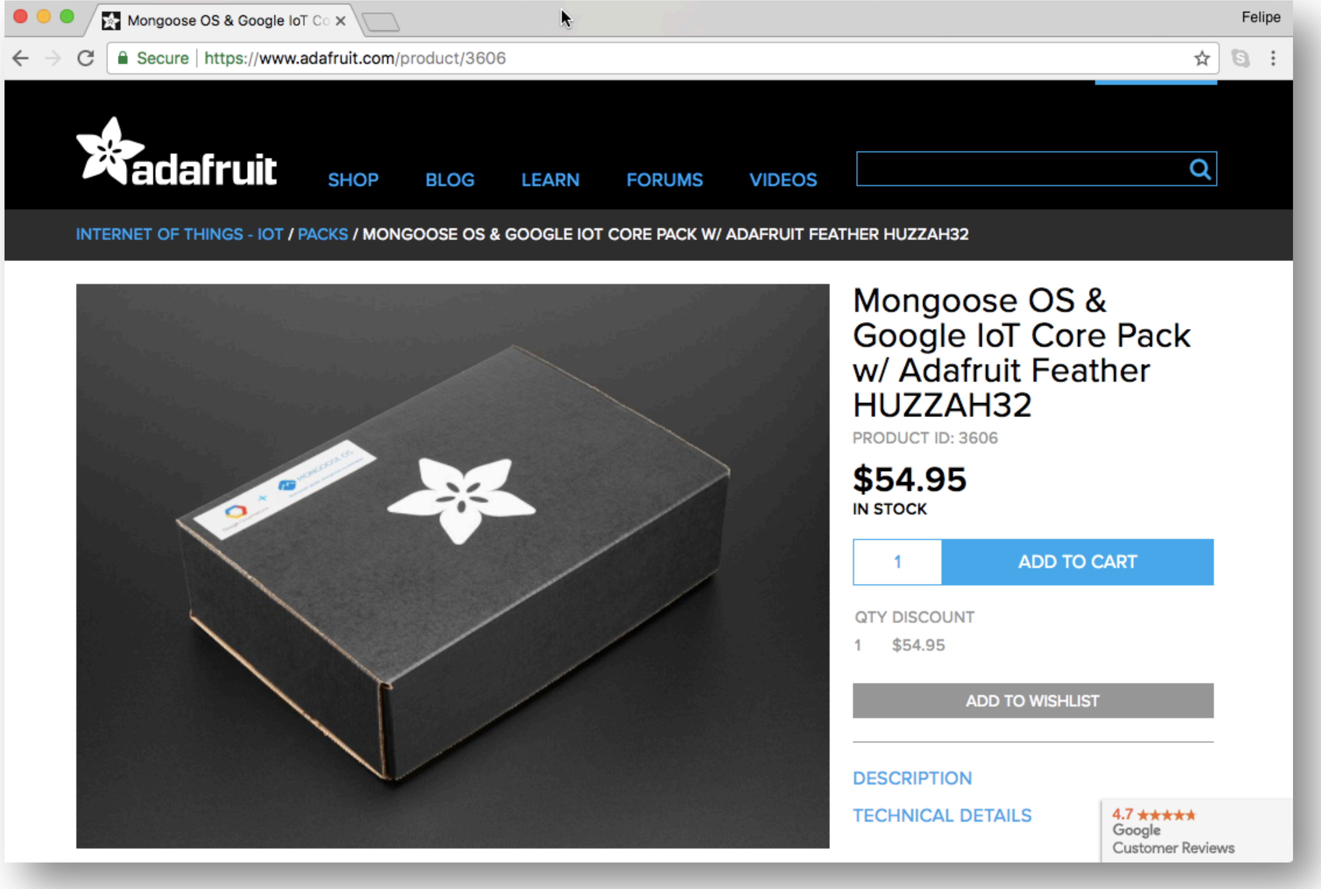
Mongoose OS

CESANTA

Bringing all devices online

- Cesanta (Dublin)
- Criado a partir da biblioteca de rede Mongoose
  - Usada por NASA, Google, Broadcomm, Samsung, Bosch …
- https://cesanta.com/
- https://cesanta.com/download.html

Introdução



Secure | https://www.adafruit.com/product/3606

INTERNET OF THINGS - IOT / PACKS / MONGOOSE OS & GOOGLE IOT CORE PACK W/ ADAFRUIT FEATHER HUZZAH32

Mongoose OS & Google IoT Core Pack w/ Adafruit Feather HUZZAH32

PRODUCT ID: 3606

$54.95

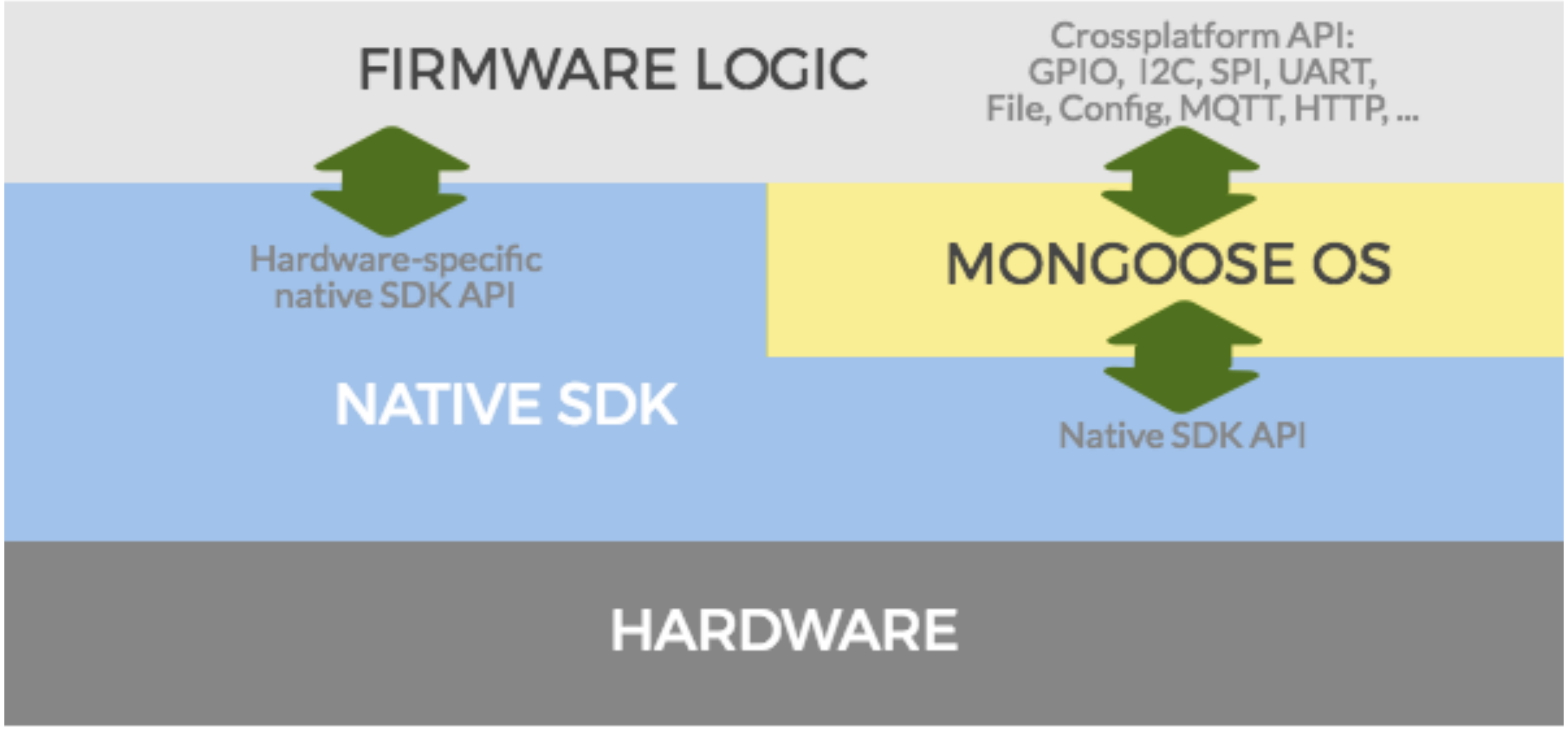
IN STOCK

QTY	DISCOUNT
1	$54.95

- https://www.adafruit.com/product/3606
- https://www.youtube.com/watch?v=zHeWNwUR2Sc

Arquitetura

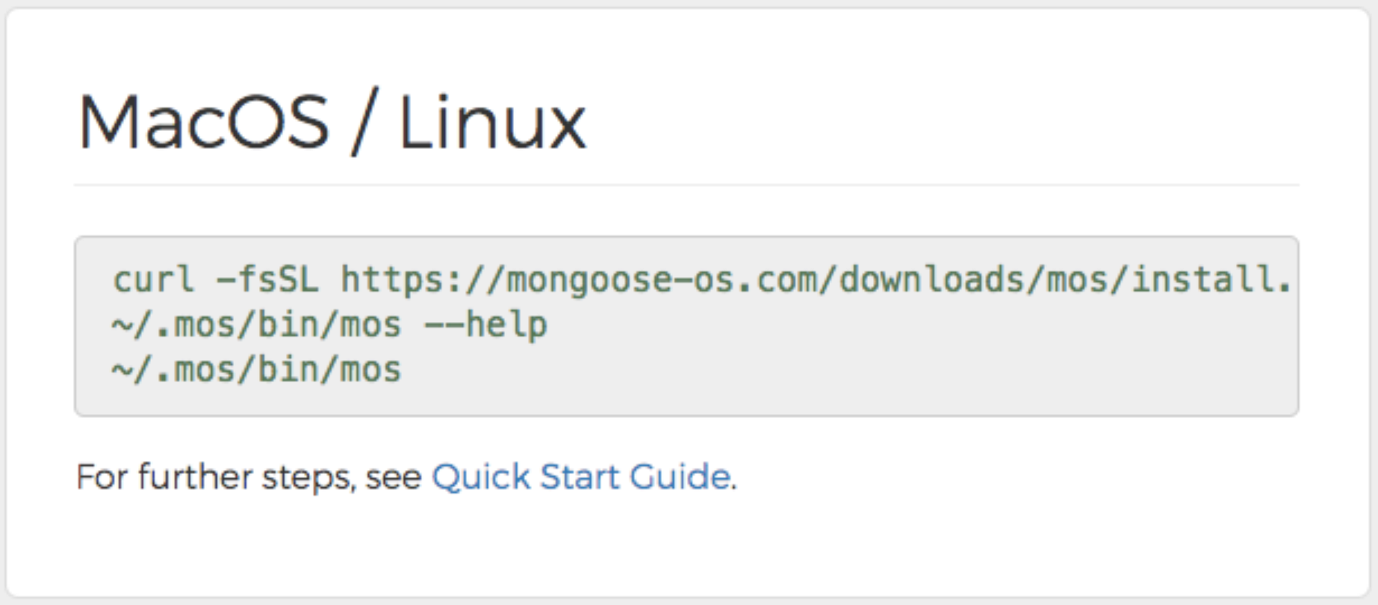
THE DEVELOPER'S CONFERENCE



- https://mongoose-os.com/docs/book/intro.html

Instalação

- https://mongoose-os.com/software.html



MacOS / Linux

```
curl -fsSL https://mongoose-os.com/downloads/mos/install.
~/.mos/bin/mos --help
~/.mos/bin/mos
```

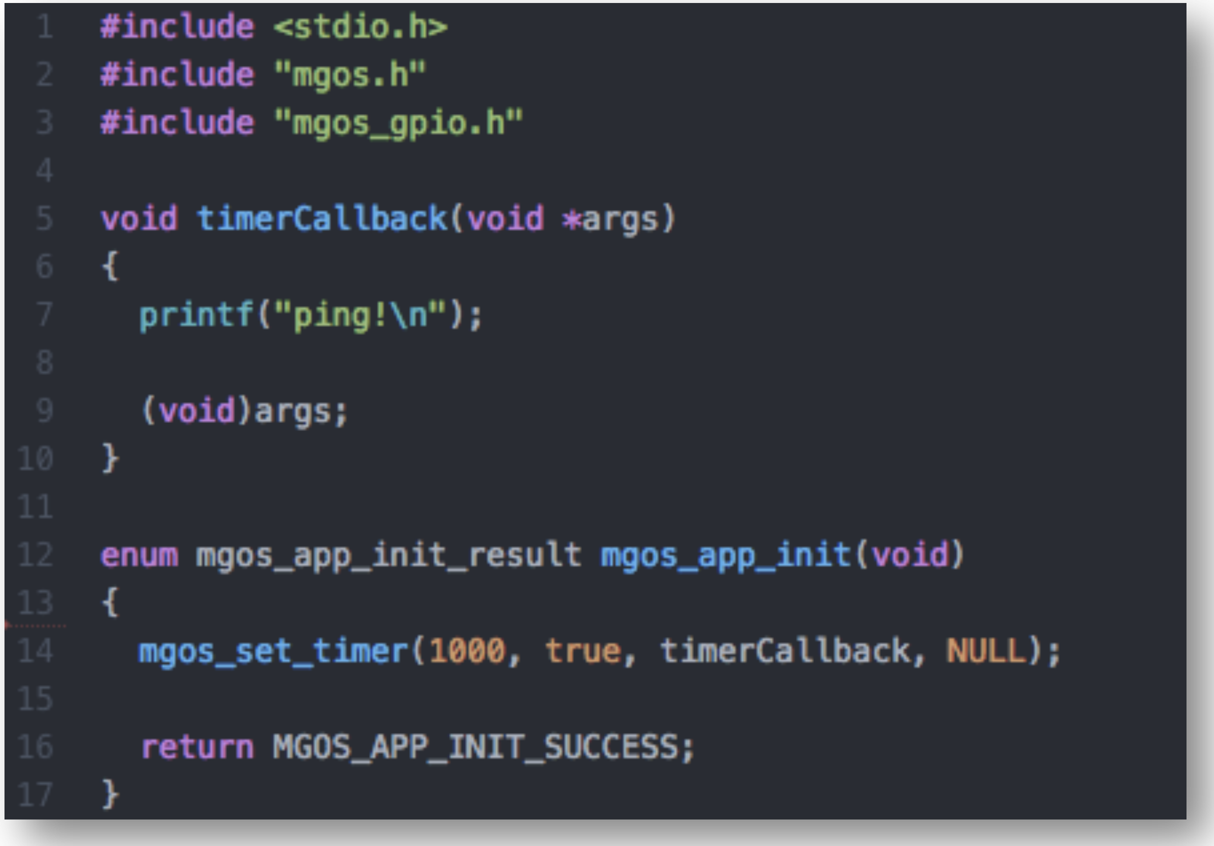
For further steps, see Quick Start Guide.

- $> export PATH=$PATH:$HOME/.mos/bin
- $> mos --help
- $> mos

Código

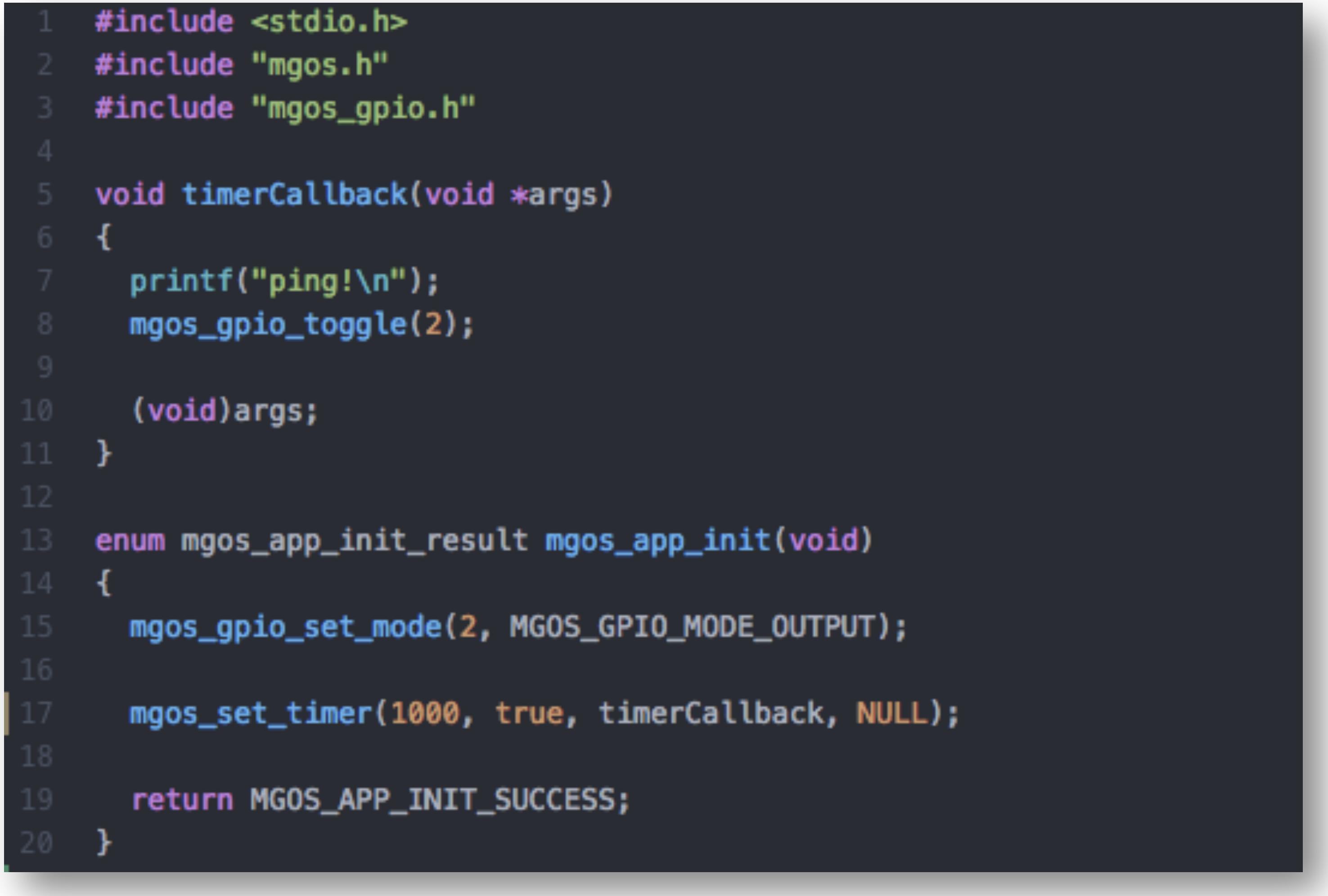
```
$> mos init [--arch esp8266]
$> mos build [--local] [--verbose]
$> mos flash
$> mos console
```

Hello World (main.c)



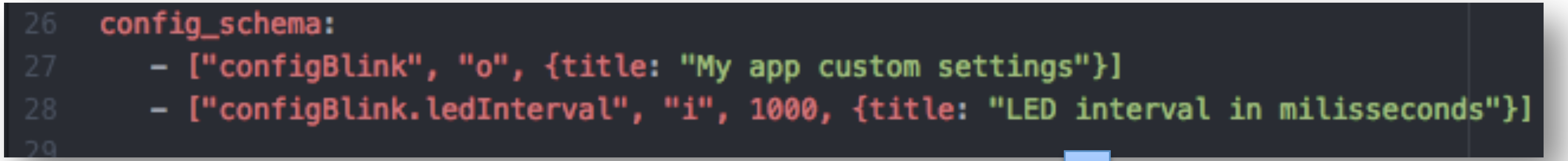
```c
#include <stdio.h>
#include "mgos.h"
#include "mgos_gpio.h"

void timerCallback(void *args)
{
  printf("ping!\n");

  (void)args;
}

enum mgos_app_init_result mgos_app_init(void)
{
  mgos_set_timer(1000, true, timerCallback, NULL);

  return MGOS_APP_INIT_SUCCESS;
}
```

Blink (main.c)

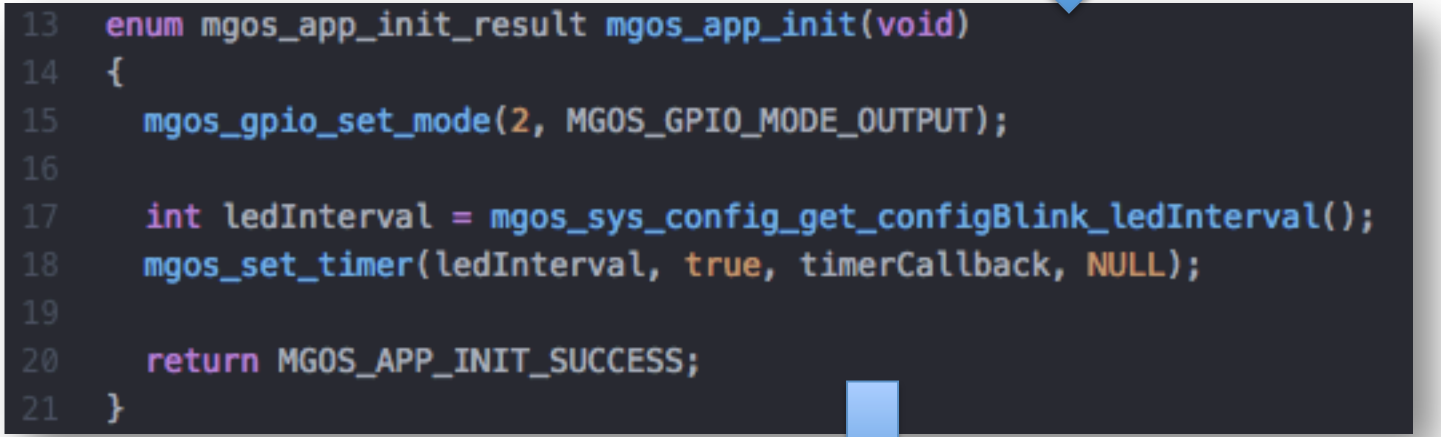


```c
#include <stdio.h>
#include "mgos.h"
#include "mgos_gpio.h"

void timerCallback(void *args)
{
  printf("ping!\n");
  mgos_gpio_toggle(2);

  (void)args;
}

enum mgos_app_init_result mgos_app_init(void)
{
  mgos_gpio_set_mode(2, MGOS_GPIO_MODE_OUTPUT);

  mgos_set_timer(1000, true, timerCallback, NULL);

  return MGOS_APP_INIT_SUCCESS;
}
```

Blink (mos.yml)



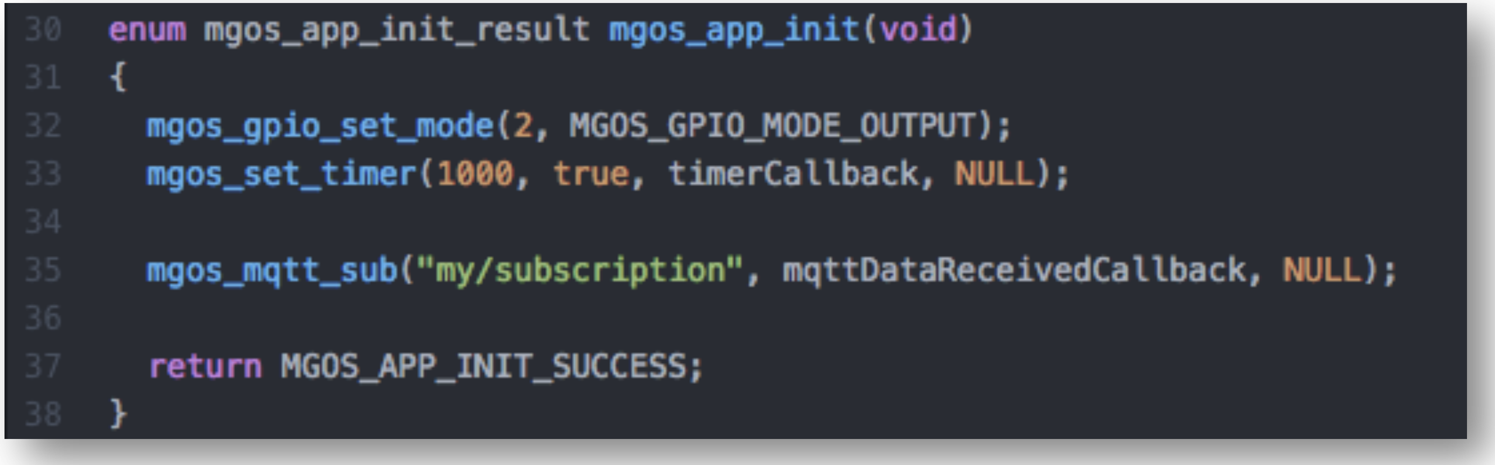
```
config_schema:
  - ["configBlink", "o", {title: "My app custom settings"}]
  - ["configBlink.ledInterval", "i", 1000, {title: "LED interval in milisseconds"}]
```



```
enum mgos_app_init_result mgos_app_init(void)
{
  mgos_gpio_set_mode(2, MGOS_GPIO_MODE_OUTPUT);

  int ledInterval = mgos_sys_config_get_configBlink_ledInterval();
  mgos_set_timer(ledInterval, true, timerCallback, NULL);

  return MGOS_APP_INIT_SUCCESS;
}
```

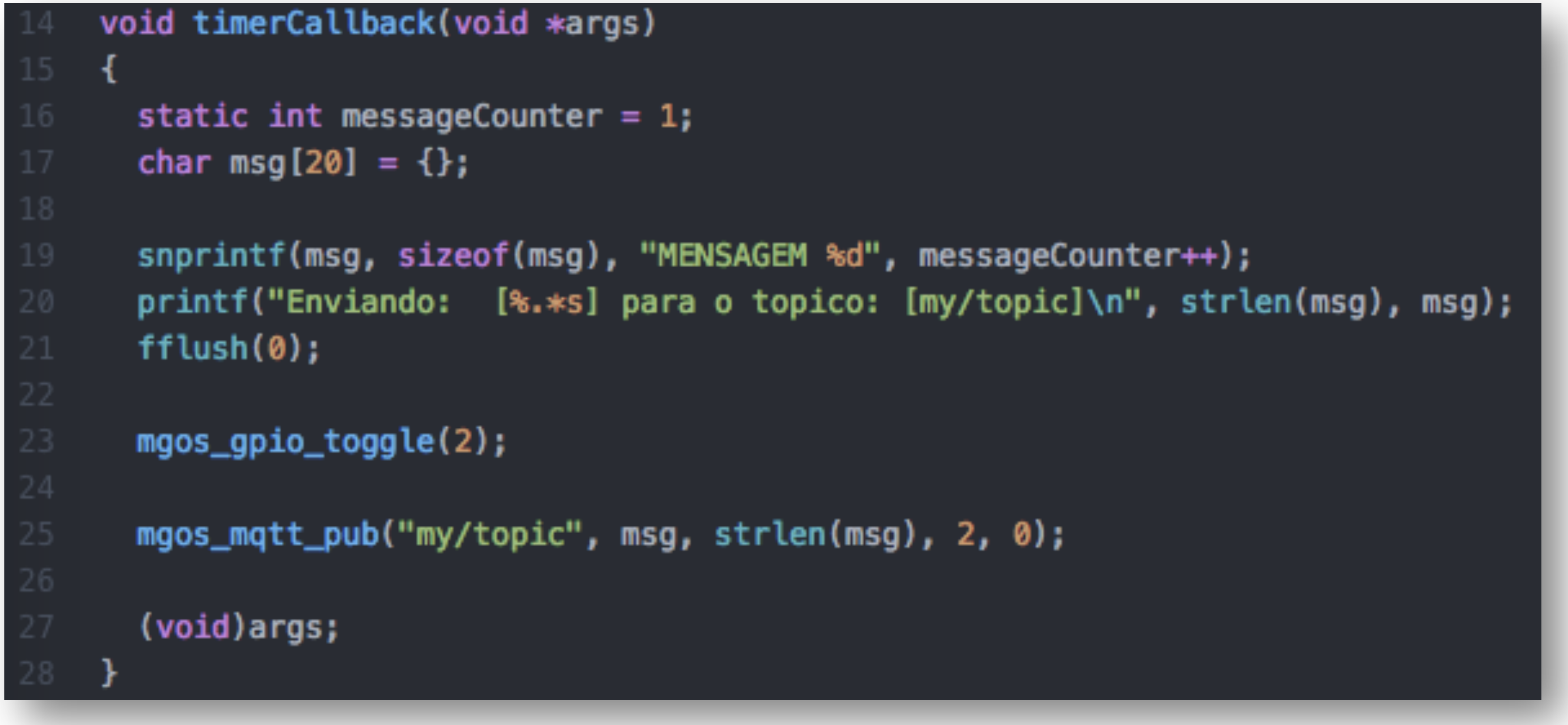
- `$> mos config-get configBlink.ledInterval`
- `$> mos config-set configBlink.ledInterval=500`

Cliente MQTT (main.c)



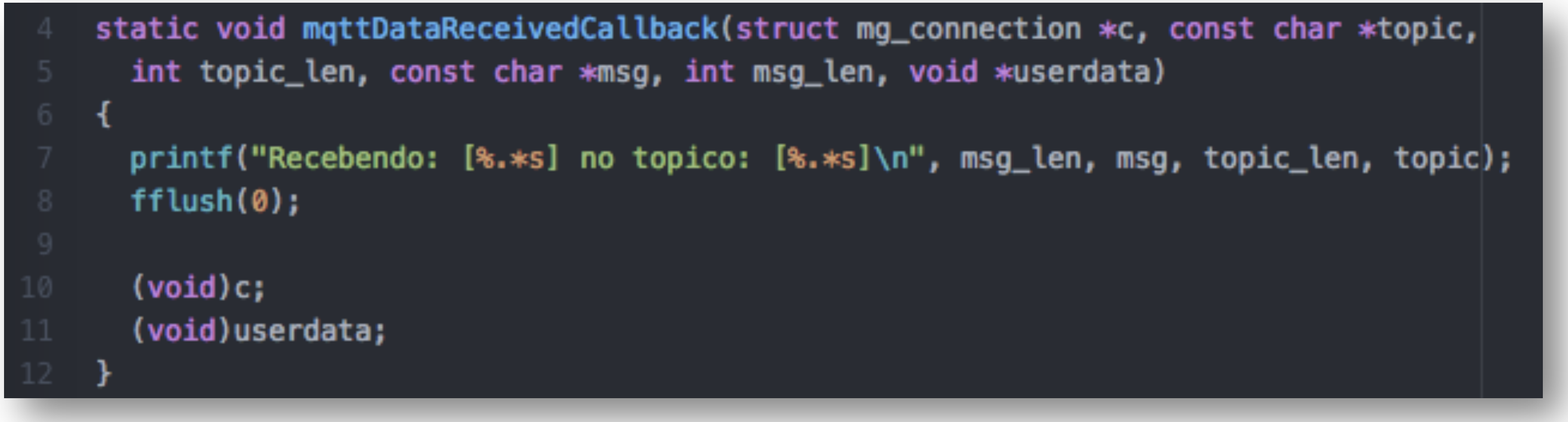
```c
enum mgos_app_init_result mgos_app_init(void)
{
  mgos_gpio_set_mode(2, MGOS_GPIO_MODE_OUTPUT);
  mgos_set_timer(1000, true, timerCallback, NULL);

  mgos_mqtt_sub("my/subscription", mqttDataReceivedCallback, NULL);

  return MGOS_APP_INIT_SUCCESS;
}
```

Cliente MQTT (main.c)



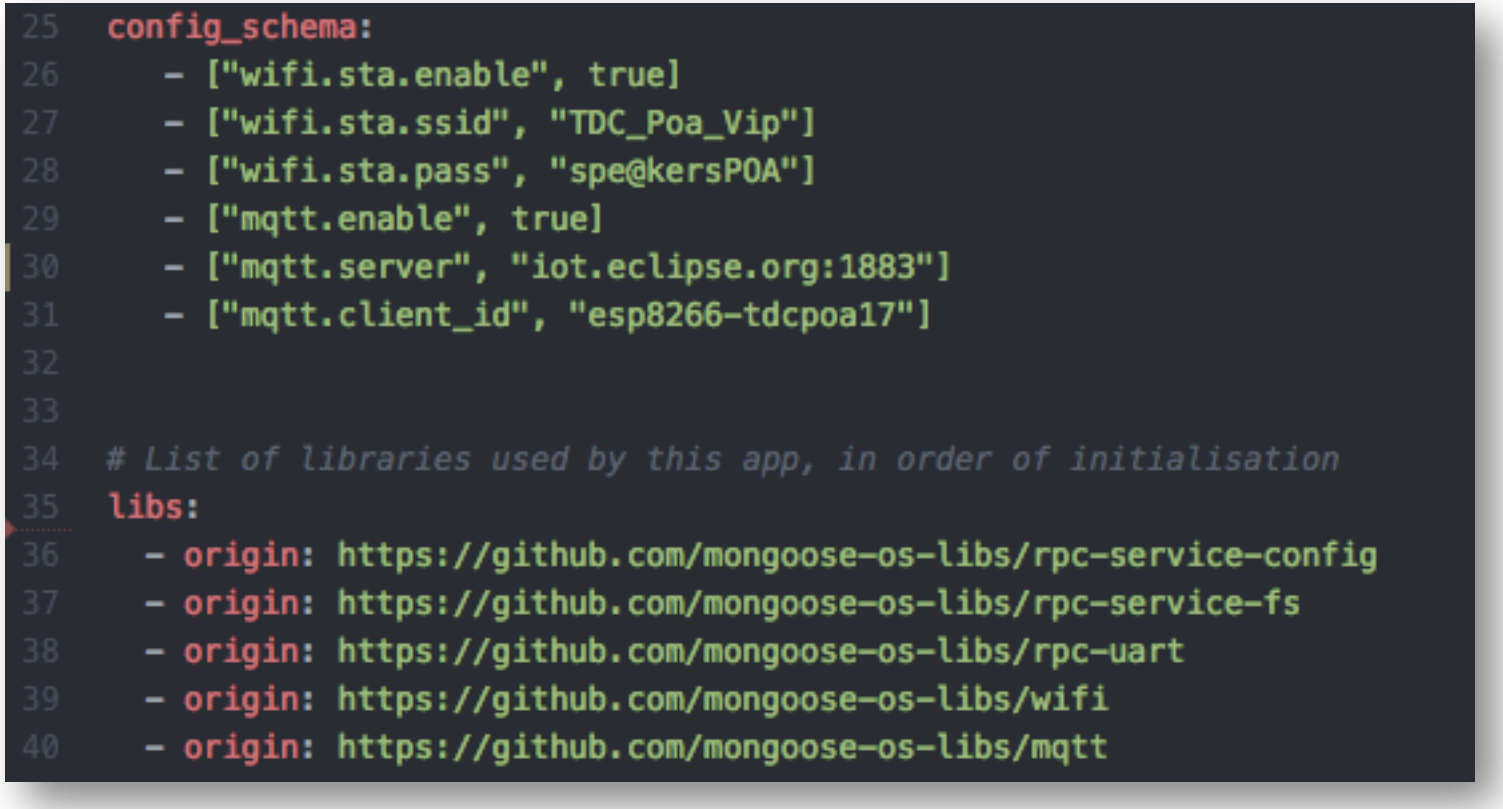
```c
void timerCallback(void *args)
{
  static int messageCounter = 1;
  char msg[20] = {};

  snprintf(msg, sizeof(msg), "MENSAGEM %d", messageCounter++);
  printf("Enviando:  [%.*s] para o topico: [my/topic]\n", strlen(msg), msg);
  fflush(0);

  mgos_gpio_toggle(2);

  mgos_mqtt_pub("my/topic", msg, strlen(msg), 2, 0);

  (void)args;
}
```

Cliente MQTT (main.c)



```c
static void mqttDataReceivedCallback(struct mg_connection *c, const char *topic,
  int topic_len, const char *msg, int msg_len, void *userdata)
{
  printf("Recebendo: [%.*s] no topico: [%.*s]\n", msg_len, msg, topic_len, topic);
  fflush(0);

  (void)c;
  (void)userdata;
}
```

Cliente MQTT (mos.yml)



```yaml
config_schema:
   - ["wifi.sta.enable", true]
   - ["wifi.sta.ssid", "TDC_Poa_Vip"]
   - ["wifi.sta.pass", "spe@kersPOA"]
   - ["mqtt.enable", true]
   - ["mqtt.server", "iot.eclipse.org:1883"]
   - ["mqtt.client_id", "esp8266-tdcpoa17"]


# List of libraries used by this app, in order of initialisation
libs:
  - origin: https://github.com/mongoose-os-libs/rpc-service-config
  - origin: https://github.com/mongoose-os-libs/rpc-service-fs
  - origin: https://github.com/mongoose-os-libs/rpc-uart
  - origin: https://github.com/mongoose-os-libs/wifi
  - origin: https://github.com/mongoose-os-libs/mqtt
```

Obrigado! ;)

Felipe Kühne

fkuhne@gmail.com

github.com/fkuhne/tdc17

Referências

THE DEVELOPER'S CONFERENCE

- https://mongoose-os.com
- https://mongoose-os.com/docs/reference/api.html
- https://mongoose-os.com/apps.html
- https://github.com/mongoose-os-apps
- https://github.com/mongoose-os-libs

- https://www.youtube.com/channel/UCZ9lQ7b-4bDbLOLpKwjpSAw

- https://github.com/mqtt/mqtt.github.io/wiki/public_brokers
- https://chrome.google.com/webstore/detail/mqttlens/hemojaaeigabkbcookmlgmdigohjobjm

Globalcode