Termuinator
(sparare ad un moscerino con un cannone)

Andrea Trentini

2023

Indice

1 Termuinator device **2**

2 Cam di lettura contacalorie **6**

3 Digressione misura consumo traffico dati VeryMobile **12**

4 Temperatura esterna **15**

5 MQTT **15**

6 Riporto letture contacalorie **15**

7 Plot finale **15**

8 Migliorie possibili **17**

Figura 1: versione originale

⟨fig:originale⟩

1 Termuinator device

- “Efficientatore” (versione full)
- “fregatore” (versione mini)
- ma anche misuratore consumo caloriferi (temperatura al calorifero è un proxy del conteggio)
- consumo ventilatore circa 50W al max
- iniziato tanti anni fa... (figura 1)
- esp8266 con relè (figura 2)
- DHT
- esphome con “thermostat” component (COOL mode)
- versione “presa comandata” (figura 3) montato al calorifero (figura 4)

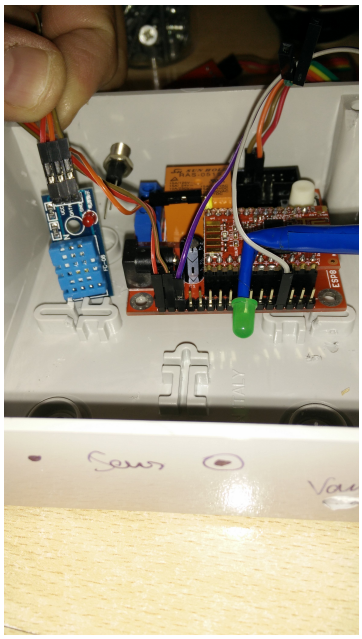

Figura 2: cuore (esp8266 Olimex)

⟨fig:cuore⟩

Figura 3: versione “presa comandata”

⟨fig:presa⟩

Sorgente esphome:

```
################################################################################
# per flashare alimentare tenendo premuto bottone, poi ricordarsi di ribootare
# https://www.olimex.com/Products/IoT/ESP8266/ESP8266-EVB/open-source-hardware
################################################################################
# provare syslog https://github.com/TheStaticTurtle/esphome_syslog

substitutions:
  devicename: termuinator-at

esphome:
  name: ${devicename}
  platform: esp8266
  board: modwifi
  project:
    name: "atrent.termuinator"
    version: "beta"
  platformio_options:
    board_build.filesystem: littlefs

<<: !include ./secrets.yaml
#<<: !include ./secrets-lab.yaml

web_server:

logger:
  level: INFO
  #level: DEBUG

#debug:
#  update_interval: 60s

mqtt:
  broker: atrent.it
  #broker: routercasa
  discovery: False
  topic_prefix: ${devicename}

sensor:
  # - platform: mqtt_subscribe
  #   name: "Watt"
  #   id: watt
  #   topic: consumo
  #   unit_of_measurement: W
  # - platform: mqtt_subscribe
  #   name: "Volt"
  #   id: volt
  #   topic: sonoff/sensor/hlw8012_voltage/state
  #   unit_of_measurement: V
  #
################################################################################
  # causa non lettura DHT?!?!!?!?!?!?
  # - platform: uptime
  #   name: "sys uptime"
################################################################################
  #
  - platform: wifi_signal
    name: "sys WiFi signal"
    update_interval: 30s
  - platform: dht
    pin: GPIO4
```

```
    temperature:
      id: temp
      name: "Temperatura"
    humidity:
      id: hum
      name: "Umidità"
    update_interval: 30s
  # - platform: debug
  #   free:
  #     name: "sys Heap Free"
  #   fragmentation:
  #     name: "sys Heap Fragmentation"
  #   block:
  #     name: "sys Heap Max Block"
  #   loop_time:
  #     name: "sys Loop Time"

binary_sensor:
  - platform: gpio
    pin:
      number: GPIO0
      inverted: true
    name: "sys Pulsante"

time:
  - platform: sntp
    id: sntp_time
    #servers: routercasa.casapastru
    timezone: Europe/Rome
    on_time_sync:
      then:
        - logger.log: "*** sync sntp"
    # on_time:
    #   - seconds: '*'
    #     then:
    #       - logger.log: "seconds"
    #       - lambda: |-
    #           ESP_LOGD("time:","%d",id(sntp_time).now().second);
    #           id(oled_display).strftime(0, 0, id(roboto), "%Y-%m-%d %H:%M", id(
        sntp_time).now());
    #           id(oled_display).printf(10, 10, id(roboto), "ciao");
    #           //id(oled_display).printf(0, 0, id(roboto), "%d:%d:%d", id(
        sntp_time).now().hour,id(sntp_time).now().minute,id(sntp_time).now().
        second);
    on_time:
      - minutes: 0
        seconds: 0
        hours: 13
        then:
          - button.press: reboot
      - minutes: 1
        seconds: 1
        hours: 1
        days_of_month: 24
        months: 2
        then:
          - lambda: |-
              auto call = id(regolatore).make_call();
              call.set_mode("COOL");
              // etc. see API reference
              call.perform();
      # - minutes: 45
```

```
      #   seconds: 30
      #   hours: 15
      #   #days_of_month: 24
      #   #months: 2
      #   then:
      #     - lambda: |-
      #         auto call = id(regolatore).make_call();
      #         call.set_mode("OFF");
      #         // etc. see API reference
      #         call.perform();


switch:
  - platform: gpio
    id: ventola
    name: "Ventola"
    pin: GPIO5

button:
  - platform: restart
    id: reboot
    name: "sys reboot"

# Example single-point configuration entry (for cooling only)
climate:
  - platform: thermostat
    id: regolatore
    name: "Regolazione"
    sensor: temp
    min_cooling_off_time: 60s
    min_cooling_run_time: 60s
    min_idle_time: 60s
    cool_action:
      - switch.turn_on: ventola
    idle_action:
      - switch.turn_off: ventola
    default_preset: Efficientamento
    preset:
      - name: Efficientamento
        default_target_temperature_high: 25 °C
        mode: "COOL"
    visual:
      min_temperature: 15
      max_temperature: 40
      temperature_step: 1
```

2 Cam di lettura contacalorie

- lettura a vista cmq difficoltosa (figura 5)
- esp32cam + usbprogrammer
- esphome
- flash led programmato tarda serata (campionamento dopo le 23)

Figura 4: montaggio finale termuinator

fig:montaggiofinale⟩

Figura 5: lettura difficoltosa

letturadifficoltosa⟩

- vibrazione dovuta al fan bloccata con gommapiuma (figura 6)
- motion (on openwrt, tuning diff!!!)
- syncthing (sync dei file)

Sorgente esphome:

```
#https://randomnerdtutorials.com/esp32-cam-ai-thinker-pinout/
#https://www.geekering.com/categories/embedded-sytems/esp32/ricardocarreira/esp32
    -cam-board-how-to-begin-and-blink-a-led/

# NOTA BENE: finché è connesso esphome in debug sulla USB non parte nulla, basta
    killare l'esphome e funziona tutto
# in generale meglio fare OTA per upload e MQTT per i log
# anche il debug via USB lo blocca

# TODO fattorizzare e fare file da includere per le def comuni

substitutions:
  devicename: cam-at

esphome:
  name: ${devicename}
  platform: esp32
  board: esp32cam
  project:
    name: "atrent.cam"
    version: "mobile"

# Enable logging
logger:
  level: INFO
  #level: DEBUG
  #level: WARN
  #level: VERBOSE
  #level: VERY_VERBOSE

#debug:

<<: !include ./secrets.yaml
```

Figura 6: Cam versione definitiva

⟨fig:cam⟩

```
mqtt:
  broker: atrent.it
  #broker: routercasa
  discovery: False
  topic_prefix: ${devicename}

# LED D0
# status_led:
#   pin:
#     number: GPIO33
#     inverted: True

web_server:
  #port: 80

light:
  - platform: monochromatic
    name: "flash"
    id: flashled
    output: ledpin

# interval:
#   - interval: 30min
#     then:
#     # - light.toggle: flashled
#     #
#     # sostituire con 'repeat'
#     # (ciclo con delay "strani") basterà per evitare Nyquist? se no mettere un
    random
#       - light.turn_on: flashled
#       - delay: 19s
#       - light.turn_off: flashled
#       - delay: 7s
#       - light.turn_on: flashled
```

```
#      - delay: 11s
#      - light.turn_off: flashled
#      - delay: 3s
#      - light.turn_on: flashled
#      - delay: 17s
#      - light.turn_off: flashled

output:
  - id: ledpin
    platform: ledc
    pin: GPIO4

time:
  - platform: sntp
    id: sntp_time
    servers: routerlab-owrt
    #servers: routercasa.casapastru
    timezone: Europe/Rome
    on_time_sync:
      then:
        - logger.log: "*** SNTP sync"
    on_time:
      - seconds: 30
        minutes: 50
        hours: 23
        then:
          - logger.log: "*** inizio ciclo led"
          - repeat:
              count: 5
              then:
              - logger.log: "--- iterazione"
              - light.turn_on: flashled
              - delay: 2s
              - light.turn_off: flashled
              - delay: 3s
      - minutes: 0
        seconds: 0
        hours: 14
        then:
          - button.press: reboot

esp32_camera:
  name: atrentcam
  external_clock:
    pin: GPIO0
    #frequency: 10MHz
  i2c_pins:
    sda: GPIO26
    scl: GPIO27
  data_pins: [GPIO5, GPIO18, GPIO19, GPIO21, GPIO36, GPIO39, GPIO34, GPIO35]
  # the order of the data_pins is significant, don't mix up the order
  vsync_pin: GPIO25
  href_pin: GPIO23
  pixel_clock_pin: GPIO22
  power_down_pin: GPIO32
  resolution: 1600x1200
  #resolution: 800x600
  #resolution: 640x480 # questo OK
  #resolution: 320x240
  #resolution: 160x120
  #idle_framerate: 0.1fps
  #max_framerate: 6fps
```

```
esp32_camera_web_server:
  - port: 8080
    mode: snapshot
  # - port: 8081
  #   mode: stream

button:
  - platform: restart
    id: reboot
    name: "system reboot"
```

Script stats.sh (stats generazione immagini, per aiutare nel tuning motion+flashing):

```
for dir in $(find -name 'term*' | cut -f-4 -d/ | sort | uniq); do
        echo $dir
        ls $dir/term* | cut -f3 -d_ | cut -f1 -d: | sort -n | uniq -c | tr -s " "
             | awk '{print $2,$1}'
done
```

```
./2023/02/14
00 110
01 106
02 111
03 110
04 126
05 123
06 110
07 139
08 145
09 152
10 130
11 168
12 131
13 142
14 190
15 118
16 147
17 133
18 114
19 101
20 108
21 102
22 105
23 108
./2023/02/15
00 118
01 116
02 112
03 109
04 103
05 124
06 118
07 125
08 166
09 143
10 142
11 146
12 120
13 135
```

```
14 229
15 125
16 205
17 126
18 115
19 113
20 117
21 115
22 105
23 100
...
./2023/02/23
00 110
01 130
02 125
03 125
04 113
05 122
06 120
07 130
08 136
09 138
10 153
11 144
12 149
13 110
14 361
15 13
16 22
17 53
18 2
23 30
./2023/02/24
00 6
03 8
07 9
08 16
09 47
10 15
11 12
12 38
13 31
14 25
15 11
16 67
17 145
18 64
23 20
...
```

3 Digressione misura consumo traffico dati VeryMobile

- VeryMobile NON ha API di consultazione traffico residuo, né un meccanismo via web, solo APP (inserire bestemmie)

- ergo ci vuole ADB + screenshot (figura 7)
- OCR (tesseract) per estrarre il dato

Script ADB screenshot:

```
#!/bin/bash

cd /Syncthing/atrent-condivisione/LabCellScreens

adb shell input keyevent 26 && adb shell input touchscreen swipe 930 880 930 380
    && adb shell input text XXXXX && adb shell input keyevent 66

adb shell monkey -p it.very.mobile -c android.intent.category.LAUNCHER 1

sleep 30

echo ...capture...
adb exec-out screencap -p > $(date +%Y-%m-%d_%H:%M).png

sleep 10

#adb shell monkey -p fr.neamar.kiss -c android.intent.category.LAUNCHER 1

# home & kill
echo ...exit...
adb shell input keyevent 4
adb shell am force-stop it.very.mobile
```

Estrazione OCR:

```
#!/bin/bash

for img in *.png; do
        OUT=$(basename "$img" .png)
        if ! test -e "$OUT.txt"; then
                tesseract "$img" $(basename "$img" .png)
        fi
done

echo data,gb >ocr.csv
grep -e "[0-9][0-9]\.[0-9]" ./*.txt | cut -f1 -d" " | cut -c3- | sed 's/\.txt:/,/
    g' | tr "_" " " | sort -n | tee -a ocr.csv

./plot.r

eog *.jpg &
```

Script R plot per ottenere figura 8:

```
#!/usr/bin/env Rscript
library(anytime)
ocr=read.csv("ocr.csv",sep=",",header=TRUE)
ocr$data <- anytime(ocr$data)
summary(ocr)
jpeg("gb.jpg",width=30,height=15,units="cm",res=300)
plot(ocr$data,ocr$g ,col="red",type="l",ylim=c(0,100))
```

Figura 7: screenshot

⟨fig:screenshot⟩

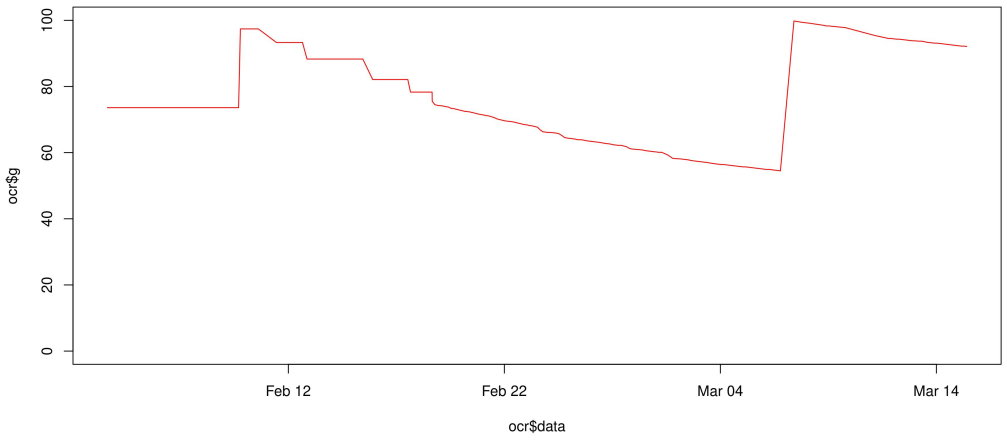

Figura 8: plot traffico

⟨fig:plottraffico⟩

4 Temperatura esterna

- raspberry pi sul balcone
- dht
- log su file
- presi via scp
- (difetto) non protetto dal sole diretto
- è su da anni

5 MQTT

- solito atrent.it
- script di cattura fatto con mosquitto_sub + qualche filtro
- save su log file CSV

6 Riporto letture contacalorie

- foglio elettronico
- interpretazione immagine a vista (OCR tentato senza successo)
- generazione CSV

7 Plot finale

Mangia un po' di CSV e plotta (figura 9):

```
library(lubridate)
library(anytime)
library(units)

################################################
t=read.csv("temps.csv",sep=" ",header=TRUE)
t$Data <- as_datetime(t$Data)

c=read.csv("cooling.csv",sep=" ",header=TRUE)
c$Data <- as_datetime(c$Data)

l=read.csv("2plot.csv",sep=",",header=TRUE)
l$timestamp <- as_datetime(l$timestamp)
```

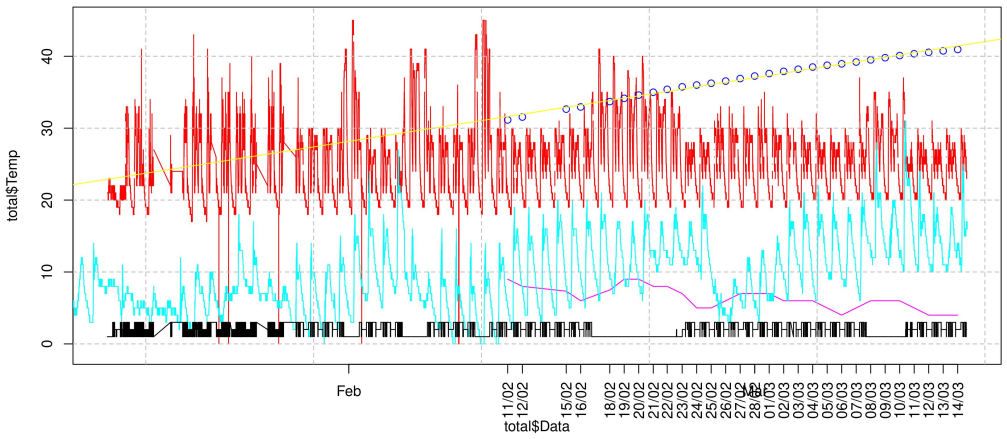

Figura 9: risultato finale

⟨fig:plot⟩

```
l$Lcorrente <- l$Lcorrente/20 # scalato
l$Lmezzanotte <- l$Lmezzanotte/15 # scalato
l$dT <- l$dT*2 # scalato

outside = read.csv("outsideTemp.csv")
outside$Data <- ymd_hm(outside$Data)

total <- merge(c,t,by="Data",all.x=TRUE)

summary(total)

jpeg("plot.jpg",width=30,height=15,units="cm",res=300)

plot(total$Data,total$Temp,col="red",type="l",ylim=c(-1,45),xtitle=NULL)

points(l$timestamp,l$Lcorrente,col="blue")
par(las=2)
axis.POSIXct(1, at=l$timestamp, labels=format(l$timestamp, "%d/%m"))

abline(lm(l$Lcorrente ~ l$timestamp),col="yellow")

lines(l$timestamp,l$Diff24,col="magenta")

lines(outside$Data,outside$temp,col="cyan")
grid(nx = NULL, ny = NULL,
lty = 2,      # Grid line type
col = "gray", # Grid line color
lwd = 1)      # Grid line width

lines(total$Data,total$Action)
```

8 Migliorie possibili

- posizionare temperatura esterna a nord
- mettere altri sensori temp in lab per confronto e check solare
- BME280 invece di DHT
- sensore temp calorifero ad appoggio (sonda), es. LM35
- wireguard e PULL (scp dei log, chirurgico) invece di PUSH (syncthing)
- nodered per processare direttamente i msg invece di passare per i csv (dallo stream mqtt)
- lettura RF del contacalorie (provata con tesista qualche anno fa, senza successo, ho la board, ma complicata)
- calcolare/prevedere direttamente i consumi in funzione della temperatura misurata al calorifero (i.e., clono la funzionalità del contacalorie: integrazione temperatura/dT)
- lettura consumo fan (magari via sonoff)
- si può scorporare le varie componenti: avere solo un termometro e solo una fan comandabile, che si parlano in MQTT, logica distribuita o centralizzata (HomeAssistant?)