

# **Gestione impianto termico con Arduino, Python e Telegram**

**Come è nato ‘SisTer’ e come si è evoluto aggiungendo tecnologia...**

# **Chi sono e il perché di questa presentazione**

Mi chiamo Riccardo Di Sarcina, ho una piccola società di consulenza e mi occupo di informatica forense, business continuity, infrastrutture di rete e sicurezza, VoIP, ecc...

Lo scopo di questa presentazione è di ‘restituire’ qualcosa a questa comunità che negli anni mi ha fornito stimoli e idee: molte le ho utilizzate in questo mio progetto e le ritroverete nelle prossime pagine.

# Oggetto della presentazione - Abstract

Con un piccolo ampliamento di casa ho rivisto la gestione dell'impianto termico (caldo/freddo) per estenderlo alle nuove zone: per l'occasione ho utilizzato dei WebRelè, Arduino e una batteria di relè per controllare le elettrovalvole di zona, ed è nato "SisTer".

Visto che "l'appetito vien mangiando" ho aggiunto un modulo Ethernet a SisTer per 'emettere' gli eventi del sistema, eventi che leggo, registro e presento grazie ad un BOT Telegram scritto in Python.

Il BOT stesso crescendo ha 'inglobato' la gestione dei WebRelè, alcuni aspetti della videosorveglianza, il controllo varchi (pedonale e carrabile), la verifica della presenza delle persone in casa e l'automatizzazione di alcuni 'scenari'.

Chissà come si evolverà in futuro...

## **Disclaimer: siamo fuori dalle mie ‘competenze’...**

Quello che ho realizzato e che vedrete presentato è frutto di curiosità e passione, di tentativi ed errori (molti), di ricerche e di copia/incolla, di ore di sonno mancate (molte...). Sono contento di non aver ‘bruciato’ nulla...

Nella presentazione semplificherò e sicuramente commetterò imprecisioni agli occhi di chi è esperto di questi argomenti: abbiate pazienza...

# Aspetti ‘idraulici’: versione iniziale



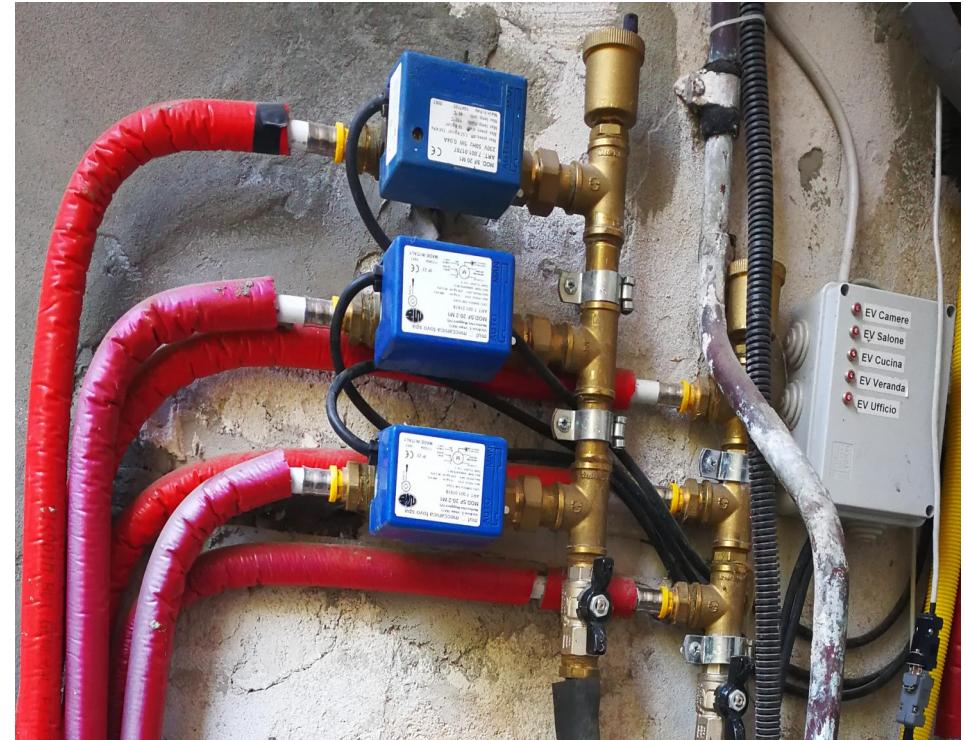
Le zone dell’impianto termico con le tubazioni e le elettrovalvole (EV) di zona di colore blu

- EV Zona giorno
- EV Zona notte
- EV Zona bagni (non in foto...)

# Aspetti ‘idraulici’: evoluzione attuale

A quanto inizialmente presente, per le tre nuove zone è stato aggiunto:

- Tre nuove EV (in blu)
- Scatola collegamenti elettrici con LED di indicazione funzionamento
- Tutte le EV se disalimentate hanno la chiusura ‘a molla’



# Fan Coil (Ventilconvettori) di zona



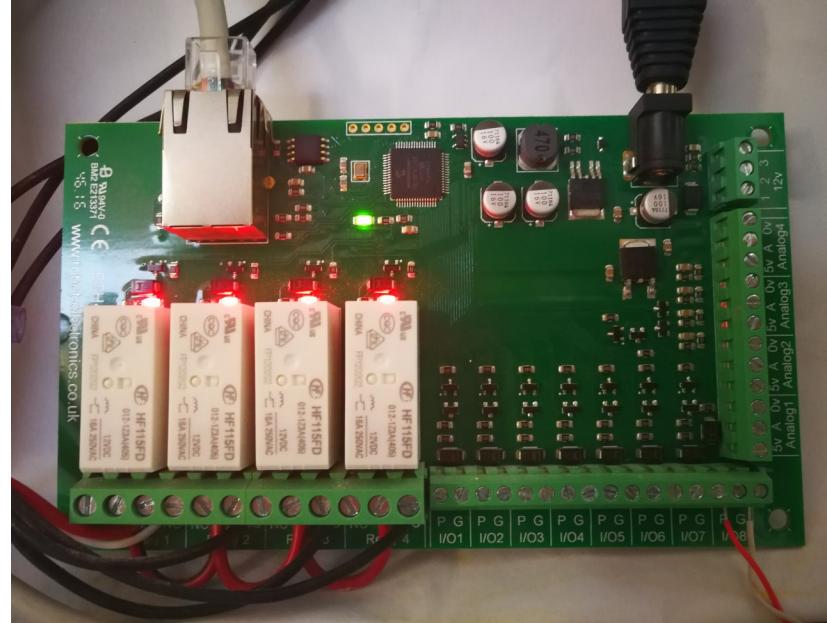
- Requisito fondamentale: I FC installati mantengono lo stato (acceso/spento) e le impostazioni (temperatura richiesta) anche se viene a mancare l'alimentazione: alla successiva riaccensione riprendono a funzionare con le impostazioni previste

# WebRelè (eth484) per controllo alimentazione FC

Col WebRelè posso accendere:

- FC Zona Giorno
- FC Zona Notte
- Prese esterne (in giardino)
- Luci ingresso

Applicazione per Android, accesso anche da remoto



ETH484\_1 @ hotsellip.it

DIG 1 - 8      DIG 9 - 16      ANALOG

Disabilita CLIMA zona giorno

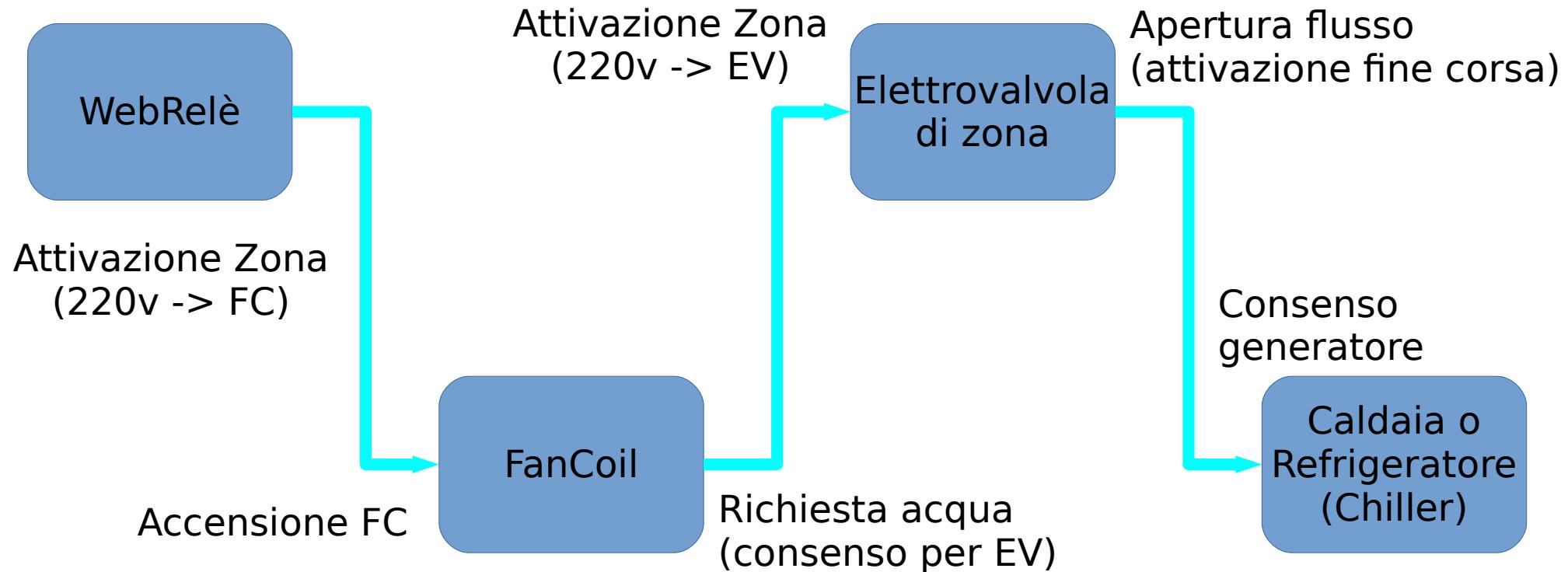
Disabilita CLIMA zona notte

Disabilita PRESE esterne

Disabilita LUCI ingresso

# Schema logico collegamenti e interazioni

## Esempio di attivazione di una zona dell'impianto



# Valutazione funzionamento sistema originale

La prima versione di gestione dell'impianto termico era stata realizzata mettendo in cascata le uscite di ogni modulo con l'alimentazione (o l'ingresso) del modulo successivo

- Funzionamento ‘stateless’
- Visione stato solo da stato relè durante funzionamento
- Sistema isolato, tranne che per attivazione via WebRelè
- Problemi emersi nei 10 anni di funzionamento
  - Il malfunzionamento di uno switch di fine corsa di una EV ha mantenuto il generatore acceso (per giorni) anche in assenza di zone attive
  - Allo spegnimento in modalità estiva la pompa del Chiller continuava a spingere mentre le EV si stavano chiudendo causando forti sollecitazioni all'impianto (e alla pompa stessa)

# Obiettivi minimi nuova versione

Possibilità di intervenire sul sistema e sul suo funzionamento

- Aggiornamenti 'software' Arduino
- Modifica parametri di funzionamento

Avere una visione anche minima dello stato del sistema

- LED sulla batteria di relè
- LED su scatola alimentazioni EV

Risoluzione problemi rilevati

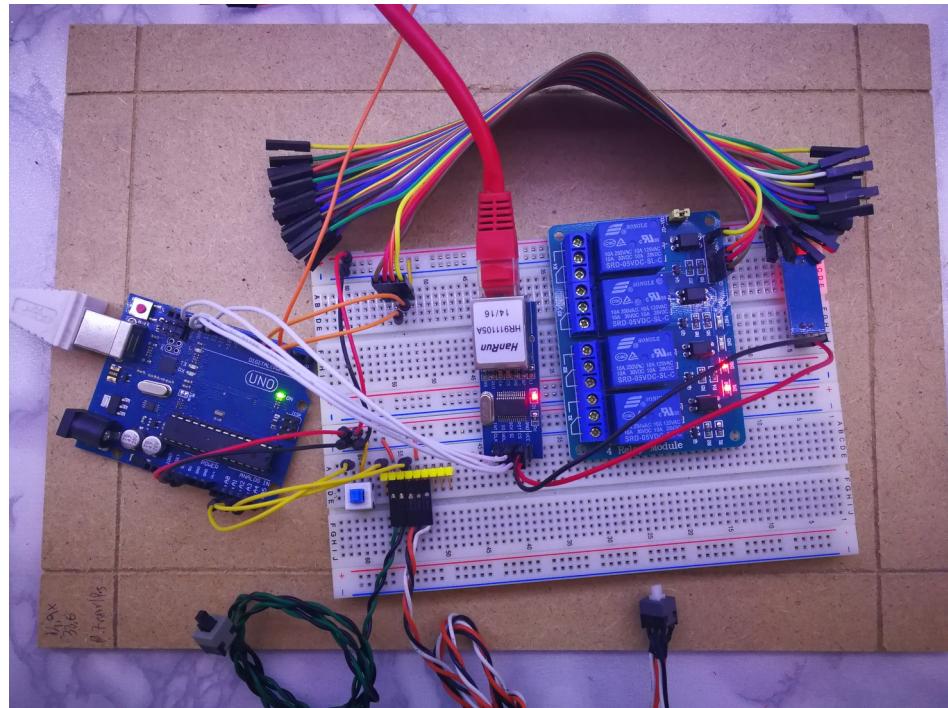
- Non dipendere necessariamente dai microswitch di fine corsa (quando EV aperta)
- Gestire temporizzazioni/ritardi per attivazione e disattivazione EV/Generatore

Apertura a estensioni/nuove funzionalità

# Pensando a SisTer (Sistema Termico)

## Inizio prototipo e test

- Dovranno essere gestiti i 3 nuovi ingressi e relative zone/EV
- Disaccoppiamento ingressi ed uscite grazie ad Arduino
- Uscita dedicata ad accensione del Chiller (qualora attivato)
- Uscita per attivazione generatore
- Pronto a nuove idee/integrazioni



# Arduino: peculiarità e caratteristiche (1)

- grande base di utilizzatori
- disponibilità di documentazione ed esempi
- molto utilizzato in progetti hardware relativamente semplici che non richiedono eccessiva potenza computazionale
- disponibilità di numerosi moduli aggiuntivi che ne estendono le funzionalità
- qualora in futuro si guastasse dovrebbe comunque essere di facile reperibilità (Arduino UNO o comunque compatibile): basterebbe così ricaricare il codice per essere nuovamente operativi...

## Arduino Uno

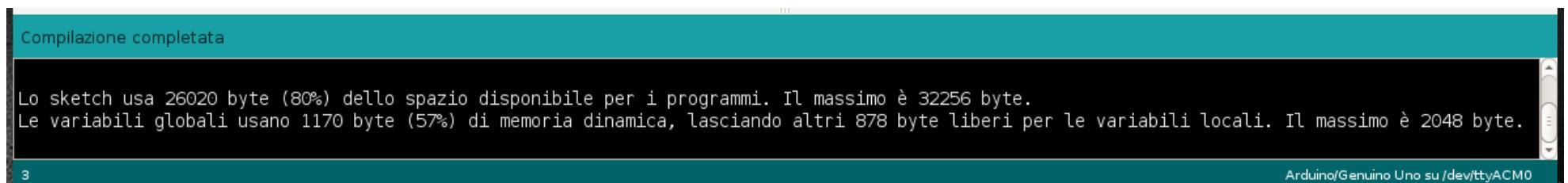
- 8-bit Microprocessor
- 32K Flash Memory
- 2K RAM
- 16 MHz Clock Speed
- Runs on 5V
- 14 Digital + 6 Analog I/O Pins



# Arduino: limitazioni

Durante la scrittura del codice più volte mi sono imbattuto nella limitata quantità di memoria di Arduino: una tecnica per limitare l'uso della preziosa RAM è quello di salvare tutte le stringhe di testo nella memoria Flash.

- L'IDE di Arduino gestisce automaticamente questa funzionalità con F()
- Es: `Serial.print(F("[Verbose] Arduino is starting..."));`



The screenshot shows the Arduino IDE interface. At the top, a teal bar displays the message "Compilazione completata". Below it, the main window shows memory usage statistics:

```
Lo sketch usa 26020 byte (80%) dello spazio disponibile per i programmi. Il massimo è 32256 byte.  
Le variabili globali usano 1170 byte (57%) di memoria dinamica, lasciando altri 878 byte liberi per le variabili locali. Il massimo è 2048 byte.
```

At the bottom left, there is a small number "3" and at the bottom right, the text "Arduino/Genuino Uno su /dev/ttyACM0".

# Arduino: Oggetto 'Digital Input'

```
#define bounce_t 200 // max ms di rimbalzo pulsante, poi input stabile

class digIN {
    byte pin;           // numero del pin di arduino
    string nome;        // descrizione testuale
    boolean input;      // ultima lettura
    unsigned long timer; // istante ultimo cambio stato (da millis...)
    boolean new_read;   // lettura che diventerà 'ufficiale' dopo "bounce_t" ms
    ...
    digIN Z1(15, "FC Camere"); // Creazione oggetto input FC Camere
    digIN Z2(16, "FC Salone"); // Creazione oggetto input FC Salone
```

# Arduino: Oggetto ‘Digital Output’

```
class digOUT {  
    byte pin;                                // numero del pin di arduino  
    String nome;                             // descrizione testuale, nome del pin  
    byte ritardo_on;                         // sec. di ritardo nell'attivazione dell'uscita  
    byte ritardo_off;                        // sec. di ritardo nella disattivazione dell'uscita  
    boolean isEnabled;                      // stato logico del relé (a meno di ritardi)  
    boolean status_req;                     // stato richiesto, diventa corrente dopo il relativo ritardo  
    unsigned long timer;                   // tempo da ultimo cambio stato (da millis...)  
    unsigned long timer_req;                // tempo da ultima richiesta cambio stato (da millis...)  
...  
digOUT R1(7, "EV Camere ", 5, 120);  
digOUT RG(2, "Generatore", 8, 1);  
digOUT RC(1, "SW Chiller", 10, 150);
```

# Arduino: not `delay()` but `millis()`

...ovvero come non FERMARE l'esecuzione del codice con `delay()` ma gestire i ritardi tramite differenze di timestamp della funzione `millis()`

- Si può gestire un ritardo come differenza di timestamps:

```
if ( ( millis() - timer_req ) / 1000 > ritardo_on) {  
    // Se è passato il ritardo previsto per l'attivazione dell'oggetto DigOUT
```

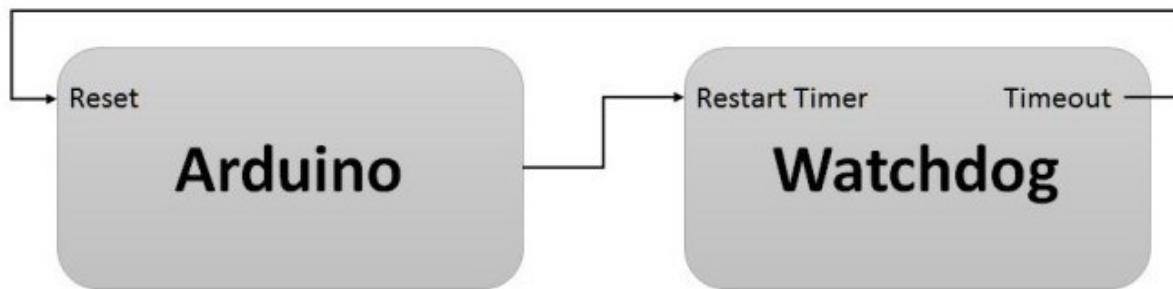
- Attualizzare i timestamp per evitare ulteriori esecuzioni dopo il ritardo previsto!

```
timer = millis();      // Attualizzo il timer (azzero l'uptime)  
timer_req = timer;    // Attualizzo il timer
```

# Arduino: impostazione del ‘watchdog’

Il watchdog è un timer, indipendente dal codice in esecuzione, che forza un reset del sistema se non riceve un segnale di “sistema ok” entro un tempo preimpostato: se il microcontrollore si blocca o il codice va in loop il watchdog, superata la soglia preimposta, effettua il reset di Arduino.

Con `wdt_reset()` azzerò il timer del watchdog.



```
#include <avr/wdt.h>

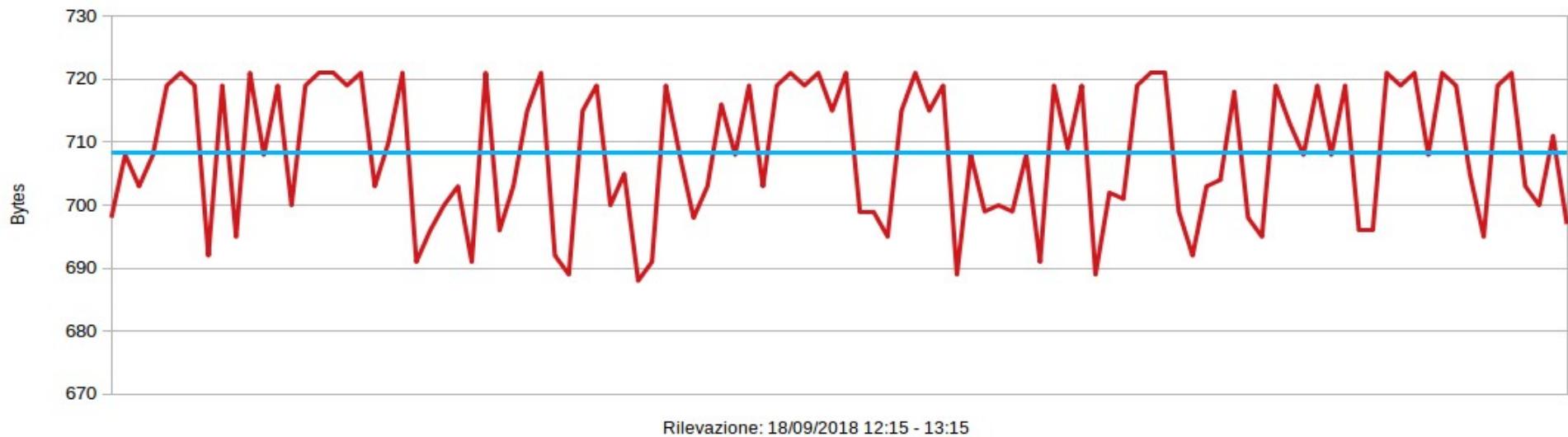
void setup() {
    // attivo con soglia a 2 Sec.
    wdt_enable(WDTO_2S);
}

void loop() {
    //eseguo qualcosa
    Delay(500);

    //resetto il watchdog
    wdt_reset();
}
```

# Arduino: analisi utilizzo RAM nel tempo

Utilizzando gli interrupt di Arduino rilevo ogni 2 secondi la RAM libera (e aggiorno minRam e maxRAM se necessario)



## [Obiettivo 1]

# Arduino: raggiunta la funzionalità del sistema

Ora l'impianto termico è gestito completamente da Arduino

- Le varie zone si attivano e disattivano come da input dei FC
- La batteria di relè è comandata tenendo conto dei ritardi preimpostati
- Le EV si attivano e disattivano secondo necessità
- Il chiller viene alimentato solo quando serve e con i ritardi on/off previsti
- Il consenso al generatore (chiller o caldaia) si attiva e disattiva correttamente

Siamo riusciti a passare l'estate 'al fresco'... :-)

# Nasce SisTer: Arduino + modulo Ethernet!

Grazie al comfort del fresco in ufficio ho pensato:  
“Dove sarà finito quel modulo Ethernet che avevo  
acquistato tempo fa? Sarebbe bello rendere  
disponibile in rete l’ stato di Arduino, senza dover  
collegare il portatile e l’USB...”

Il modulo è ENC28J60 (costo 3/4€), si collega tramite  
BUS I<sup>2</sup>C (4 PIN dedicati) e l’IDE di Arduino ha le librerie  
necessarie per il suo funzionamento (es. UIPEthernet)

Ho implementato un Telnet Server da usare come  
‘mirror’ della seriale per l’output degli eventi.

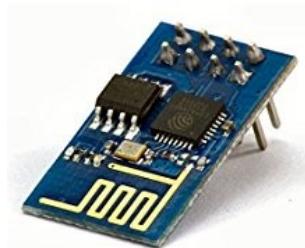
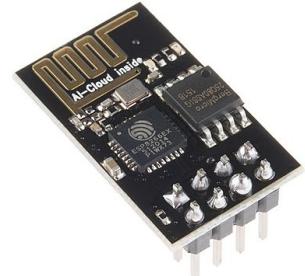


# SisTer: possibile alternativa con modulo Wi-Fi

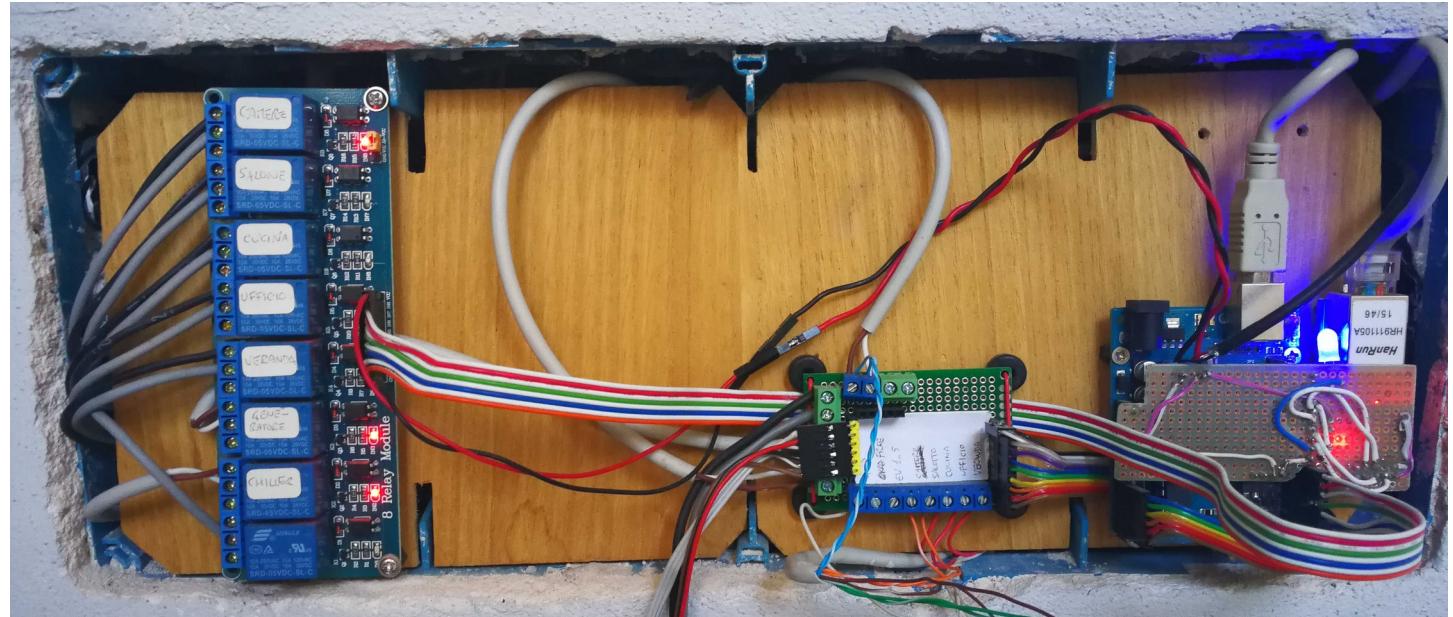
Come alternativa alla connessione Ethernet potevo utilizzare un ESP8266, microcontrollore a basso costo (circa 7/8€) e dotato di interfaccia Wi-Fi nativa.

- Microcontrollore autonomo e programmabile
- Wi-Fi ok, ma deve essere sotto copertura di un AP
- Alimentazione a 3,3v (max tensione anche su I/O)

Il collegamento sarebbe stato più semplice (solo i PIN 0 e 1 della seriale di Arduino, che sarebbe stato sgravato anche del Telnet Server) ma ho preferito non introdurre un ulteriore elemento, l'Access Point, che se non disponibile avrebbe isolato SisTer



# SisTer: prototipo in test



# SisTer: esempio di 'missione eventi'

```
2018-10-16 21:19:08 [Info] IN ;FC Salone ;Switch CHIUSO; (Uptime APERTO: 0s)
2018-10-16 21:19:14 [Info] OUT;EV Salone ;Rele' CHIUSO; (Uptime APERTO: 322s)
2018-10-16 21:19:22 [Info] OUT;Generatore;Rele' CHIUSO; (Uptime APERTO: 360s)
2018-10-16 21:19:24 [Info] OUT;SW Chiller;Rele' CHIUSO; (Uptime APERTO: 213s)
2018-10-16 21:21:46 [Info] IN ;FC Salone ;Switch APERTO; (Uptime CHIUSO: 158s)
2018-10-16 21:21:48 [Info] OUT;Generatore;Rele' APERTO; (Uptime CHIUSO: 145s)
2018-10-16 21:22:18 [Info] OUT;EV Salone ;Rele' APERTO; (Uptime CHIUSO: 183s)
2018-10-16 21:22:42 [Info] IN ;FC Camere ;Switch CHIUSO; (Uptime APERTO: 625s)
2018-10-16 21:22:48 [Info] OUT;EV Camere ;Rele' CHIUSO; (Uptime APERTO: 545s)
2018-10-16 21:22:56 [Info] OUT;Generatore;Rele' CHIUSO; (Uptime APERTO: 67s)
2018-10-16 21:24:12 [Info] IN ;FC Camere ;Switch APERTO; (Uptime CHIUSO: 86s)
2018-10-16 21:24:12 [Info] OUT;Generatore;Rele' APERTO; (Uptime CHIUSO: 73s)
2018-10-16 21:26:10 [Info] OUT;EV Camere ;Rele' APERTO; (Uptime CHIUSO: 201s)
2018-10-16 21:26:40 [Info] OUT;SW Chiller;Rele' APERTO; (Uptime CHIUSO: 434s)
```

# SisTer: analisi dati funzionamento e anomalia!!! (se non puoi misurararlo sei come John Snow...)

Analizzando il log del sistema relativo alla notte del 20 agosto 2018 e aggregandone i dati ho notato numerosi eventi emessi dai FC delle camere:  
412 fra ON e OFF, **TROPPI!!!**

	Durata media ON	Durata media OFF	Totale attivazioni (ON + OFF)
Fc Camere	72 sec.	Oltre 300 sec.	<b>412 (&gt;6.000 cicli/mese)</b>

Per questo eccesso di attivazioni e disattivazioni probabilmente si è guastata l'EV che ho dovuto sostituire...

Ho fortemente ridotto i cicli della EV della zona camere impostando il ritardo di disattivazione della 'EV Camere' a 120s, riducendo così le aperture e chiusura a qualche decina al giorno.

## [Obiettivo 2]

### Telegram come 'frontend' di SisTer

Telegram è un servizio di messaggistica istantanea veloce e sicuro che si basa su un protocollo open source: è disponibile l'applicazione nativa per gli smartphone e sono supportati anche tutti gli OS/Desktop attuali.

Le API permettono la creazione di programmi (BOT) che possono interagire direttamente con Telegram per l'invio e la ricezione di messaggi, immagini, files, ecc...

Se non l'avete mai utilizzato, provatelo...



# Telegram: vantaggi e svantaggi

## PRO:

- Autenticazione, sicurezza e trasporto già 'forniti' dal sistema
- BOT di 'facile' implementazione grazie anche a librerie (es. Telepot)
- Client multi-piattaforma e ad accesso multiplo contemporaneo



## CONS:

- È un servizio di messaggistica, quindi interfaccia utente poco personalizzabile
- Client open source ma server 'chiusi', potrebbero chiudere il servizio?
- Il down della piattaforma Telegram (raro...) causa l'inutilizzabilità del sistema

# BOT Telegram: integrazione con SisTer

Il BOT si presenta come un client Telnet e rimane in ascolto degli eventi emessi da SisTer: a tutti gli eventi viene apposto un timestamp prima di salvarli in un file di LOG.

Inizialmente riceveva solo gli eventi, poi ho introdotto un ‘keep-alive’ per mantenere attiva la connessione, o qualora caduta, per accorgersene e ristabilirla: ogni 30 secondi invio ‘ping’ e mi aspetto ‘PONG’ come risposta.

La risposta si è poi evoluta integrando altre informazioni, come l’uptime, la ram libera e l’uso min/max di memoria dall’avvio di Arduino.

Esempio di risposta al “ping”

“PONG, { 'SisTer' : [2656878, 719, 692, 745] }”

# BOT: come e di cosa parla con SisTer...

Il BOT, tramite la connessione Telnet, interagisce con SisTer: ecco i comandi disponibili

```
if      (inData == F("PING"))          testo = quickStatus(true);
else if (inData == F("STATO") || inData == F("S")) stampaStato();
else if (inData == F("BANNER") || inData == F("B")) testo = getBanner();
else if (inData == F("RAW") || inData == F("R")) stampaRAW();
else if (inData == F("OUTDUMP") || inData == F("O")) DumpOUTState();
else if (inData == F("FREE") || inData == F("F")) testo = getRam();
else if (inData == F("UPTIME") || inData == F("U")) testo = getUptime(millis());
else if (inData == F("BYE"))
else if (inData == F("RESET"))
asm volatile (" jmp 0");

#endif VERBOSE
else if (inData == F("VERBON")) { isVerbose = true; testo = F("[Info] Attivazione livello di VERBOSE\n"); }
else if (inData == F("VERBOFF")) { isVerbose = false; testo = F("[Info] DISattivaz. livello di VERBOSE\n"); }
#endif
```

## [Obiettivo 3]

# ...e i device/sistemi esistenti non li integriamo?

("Un BOT per domarli tutti"... o era un anello?)

## Dispositivi in rete integrati nel BOT

- **ETH484 (WebRelè):** API (con esempi Python) per completa gestione relè e I/O
- **Videocitofoni IP:** API per impulso apertura cancello e per 'scatto foto'
- **Telecamere IP:** URL specifiche per ottenere un fotogramma in diretta
- **ZoneMinder:** URL per ottenere un fotogramma in diretta

## Altri servizi integrati nel BOT

- Accesso giornaliero a siti internet (meteo) per reperimento dell'orario del tramonto, utilizzato negli scenari per decidere se accendere le luci esterne

# Aggiungendo funzionalità arriviamo a ‘DomotiCASA’...

Scenari ed automatismi ‘intelligenti’

- Le azioni previste sono eseguite in maniera trasparente anche se coinvolgono oggetti distinti
- L’istante della richiesta è importante per la scelta delle azioni da effettuare
  - Se sto rientrando a casa dopo il tramonto accendo le luci del camminamento dal parcheggio: all’arrivo troverò tutto illuminato...
  - Se rientro a casa dopo cena viene attivato SisTer nella zona notte, altrimenti vengono attivate le zone giorno (salone e cucina)

# Proattività in DomotiCASA

Grazie a un periodico ‘ping’ ai devices o ai cellulari personali è possibile identificare eventuali situazioni ‘anomale’

- Se nessuno è presente in casa e SisTer è attivo in qualche zona, è possibile essere avvisati che dopo 15 minuti l'impianto termico verrà disattivato automaticamente
- Se esco dall'ufficio spegnendo il PC (quindi la rilevazione via Ping avrà esito negativo) ma lasciando attivo il FC, dopo 5 minuti mi viene chiesto via messaggio Telegram se desidero disattivare la zona
- Arduino dovrebbe essere sempre disponibile, grazie anche al watchdog impostato, ma se non lo fosse per più di qualche minuto verrei subito avvisato del problema

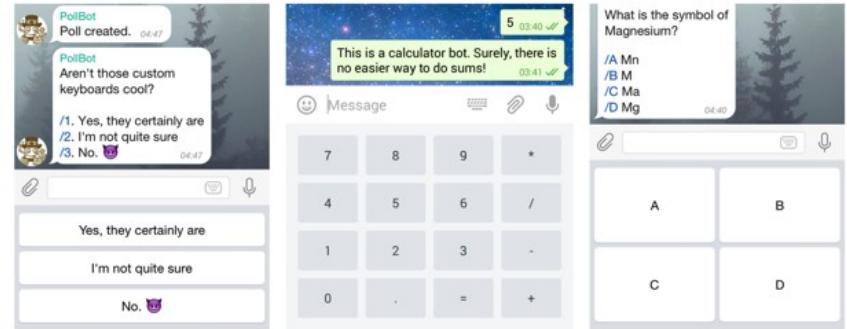
Niente di indispensabile, ma semplici ‘reminder’...

# BOT: interazione con le tastiere

L'interazione con l'utente avviene grazie alle tastiere di Telegram. Ogni pulsante della tastiera rispecchia lo stato attuale dell'oggetto che viene rappresentato: nel caso di SisTer lo stato viene richiesto col comando 'OUTDUMP' prima della generazione della tastiera

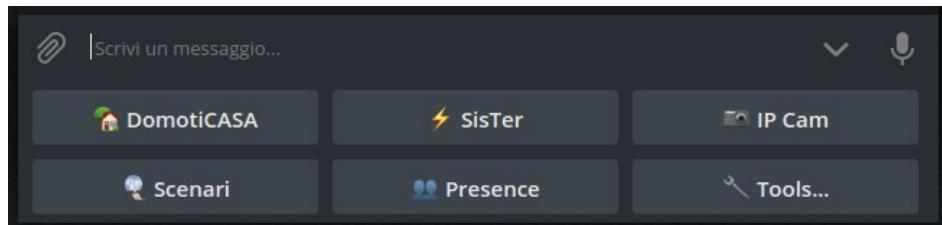
Telegram offre due tipi di tastiere:

- normali (immagine sopra)
- inline (immagine sotto)

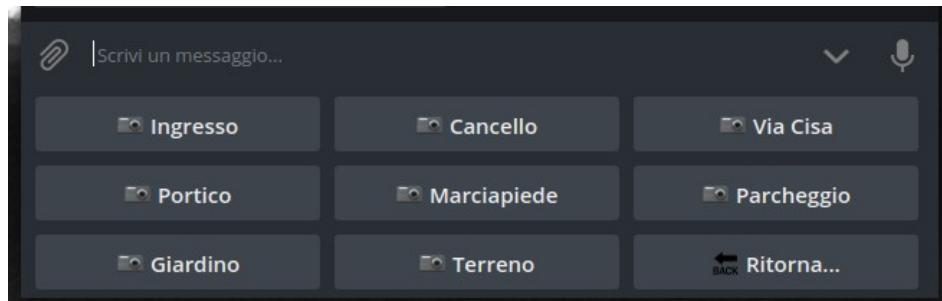


# BOT: tastiere per accesso funzionalità

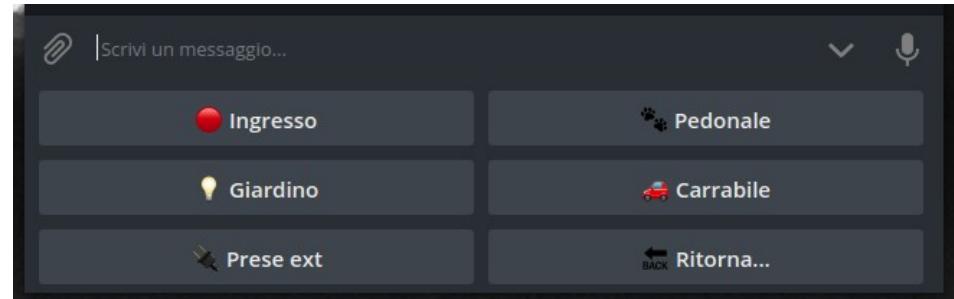
Menù principale



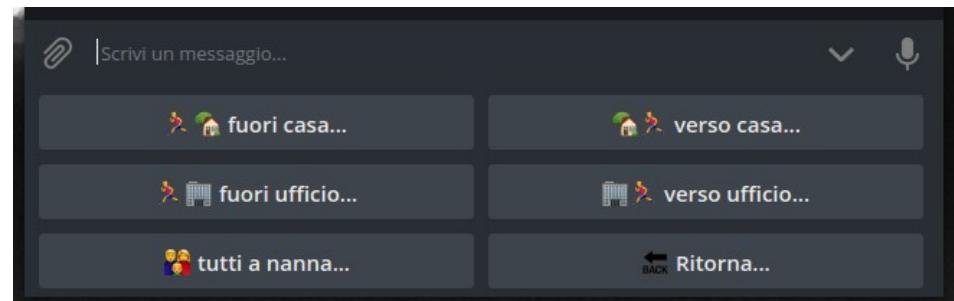
Sottomenù IP Cam



Sottomenù DomotiCASA



Sottomenù Scenari



# **BOT: dove viene eseguito e perché?**

Il BOT è un programma scritto in Python che ha bisogno di essere sempre in esecuzione: nel mio caso è in funzione in un ‘container Docker’ attivo nel NAS Qnap presente nella rete interna.

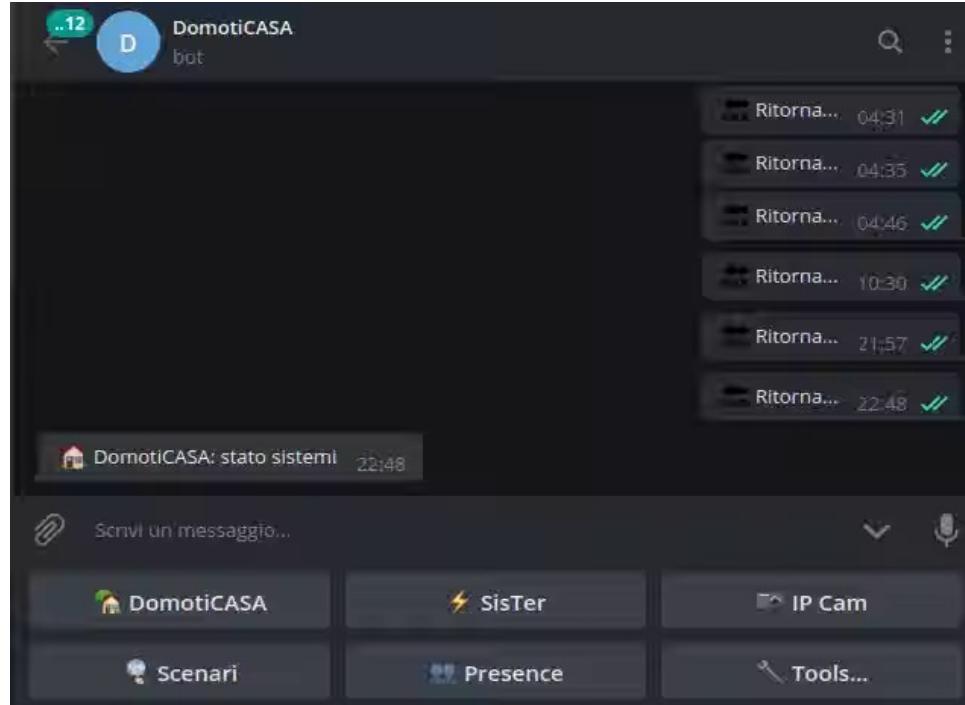
## **Altre opzioni per mantenere in esecuzione un BOT**

- Un server, un RaspberryPI o un PC in rete che rimane sempre acceso (Linux, Win o MacOS)
- Uno dei servizi disponibili in Cloud, spesso gratuiti e con numerose possibilità di estensione

Avere il BOT in esecuzione nella rete interna ha come vantaggio che i dispositivi con i quali interagisce sono direttamente raggiungibili.

**NESSUN port forward per ‘esporre’ qualcosa su internet**

# Demo di SisTer/DomotuCASA - Video



# E se Telegram/internet è down? Abbiamo un piano B?

“Nessun piano di battaglia sopravvive all'impatto col nemico”

Colin Powell, 65º Segretario di Stato degli Stati Uniti sotto il Presidente George W. Bush

Le funzionalità sono sicuramente fortemente ridotte, ma SisTer è di fatto un sistema autonomo che non necessita del BOT o della connettività ad internet per il pieno funzionamento.



# E se Telegram è down + Arduino è guasto? Piano C!

“La fortuna è cieca, ma la sfiga ci vede benissimo”

Roberto “Freak” Antoni, fondatore e leader degli Skiantos

Le EV installate (con ritorno a molla) hanno una levetta che, sollevata, può essere bloccata (EV in modalità aperta), rendendo l'impianto un ‘normalissimo’ impianto a zona singola.



# Possibili nuove evoluzioni

Il sistema ‘DomotiCASA’ crescerà, per esempio con:

- mail giornaliera/settimanale con statistiche di SisTer
- allarme domestico perimetrale
- rilevazione consumi in tempo reale (e allarmi/azioni)
- distribuzione Voucher per accesso WiFi ospiti
- ...

Suggerimenti?

# Grazie per l'attenzione...

## Domande?

### Riferimenti:

 Riccardo.Di.Sarcina@gmail.com  
 @RDiSa

Slide rilasciate sotto  
Licenza Creative Commons

