Sommario

1. Diagramma a blocchi del sistema 4
2. Preparazione del webserver 5
3. Installazione e programmazione dei server Arduino 5
3.1. Server1 6
3.2. Server2 9
4. Collegamento dei relè su Arduino 11
4.1. Collegamenti per Server1 11
4.2. Collegamenti per Server2 11
5. Software di gestione 11
5.1. Installazione del software 12
6. Test del sistema 13
7. Integrazione dei server sull'impianto 14
8. Costi 16
9. Licenza d'uso 16

Rev. 2.0.0 del 28/05/2016

**SISTEMA GESTIONE DOMOTICA **

__ ** COME FARE ** __

.



__ ** Argomenti della presente guida ** __

Lo scopo di questa guida è di implementare un sistema domotico a basso costo che ci permetta di interagire con alcuni componenti della nostra casa in modo da poterli gestire più facilmente anche lontani da casa e di sapere come vanno i consumi di casa.

Il sistema che implementeremo interagirà con i seguenti componenti:

- Gestione delle tende da sole
- Apertura e chiusura del cancello carraio
- Apertura e chiusura del portone garage
- Telegestione di un termostato GSM

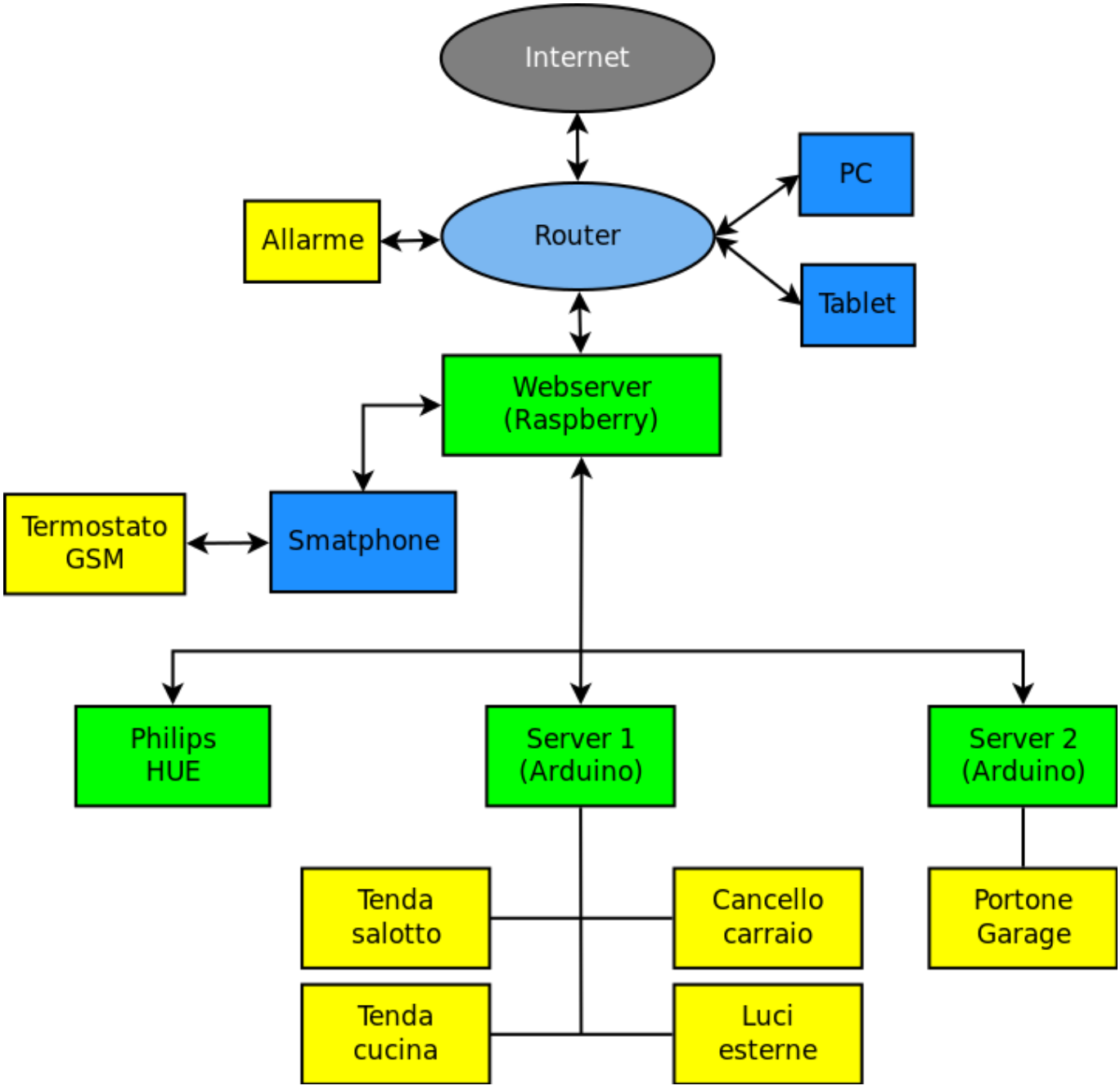
per quanto riguarda la registrazione dei consumi e della produzione di energia si rimanda alle apposite guide “Raspberry come datalogger” e “ OperEnergyMonitor su raspberryPi”.

Sul diagramma a blocchi che segue sono presenti due opzioni per la gestione delle luci, una è implementata tramite un sistema “Philips hue”, si compra già belle e fatto ma è troppo costoso,

e una seconda che si integra con il sistema e l'impianto elettrico di casa.

1. Diagramma a blocchi del sistema

Nel diagramma che segue si schematizza la struttura del sistema



Il sistema è formato da un web server (RaspberryPi) su cui risiede il software che fa da front-end per interagire con i server periferici (Arduino) che a loro volta si occupano di attivare e/o disattivare le utenze.

Sul grafico il termostato GSM è gestito a parte in quanto già di suo è autosufficiente, quello che ho fatto è implementare una interfaccia grafica per facilitare la gestione in quanto il suo sistema prevedeva unicamente l'invio di SMS.

La scelta di usare due server Arduino e non uno soltanto è legato unicamente al problema della distanza tra il quadro di connessioni generale e il motore del portone del garage, quando un anno fa ho implementato il sistema ho provato per settimane a cercare un sistema wireless per attivare il relè ma i sistemi presenti allora risultavano molto costosi, quindi decisi di utilizzare una powerline e un secondo Arduino, ciò non toglie che sarebbe possibile usarne uno solo. Ad oggi si trovano in commercio per pochi euro (3.5 per la precisione) delle

schedine tipo le Wemos D1 mini, che sono degli Arduino “light” con scheda WiFi, assolutamente da prendere in considerazione.

2. Preparazione del webserver

Come detto la scelta per gestire il web server è ricaduta sul Raspberry, costa poco circa 25€, consuma pochissimo e oramai è facilissimo da usare.

Procedere a caricare l'immagine Raspbian Jessie e a configurare il webserver seguendo la guida che trovate al seguente LINK

Terminata l'installazione ci troveremo il webserver funzionante in /var/www/ e in questa cartella andremo a installare il nostro software.

3. Installazione e programmazione dei server Arduino

Per questi server abbiamo bisogno di due Arduino Uno e di due ethernet shield io ho usato la WZ5100




per poter contenere al massimo i costi ci dobbiamo rassegnare ad utilizzare per entrambe dei cloni di produzione cinese, che comunque funzionano benissimo.

Cominciamo con l'accoppiare la WZ5100 con l'Arduino, fate attenzione ad allineare bene i pin sopratutto quelli del connettore centrale sul posteriore.

Se avete optato per i cloni cinesi dovete fare attenzione che alcune shield hanno i connettori corti e la scheda potrebbe andare in contatto con il case del connettore della porta USB dell'Arduino generando dei problemi, se è il vostro caso basta isolarla con un po di nastro.



Passiamo ora a caricare lo sketch, si suppone che siate in grado di scrivere un programma e di caricarlo, nel caso in internet ci sono un sacco di guide.

Se non volete scrivere tutti gli sketch li potete scaricare dal mio sito al seguente link Domotica

Nel nostro caso abbiamo due server:

- Server1 ⇒ Gestione tende, cancello carraio e luci
- Server2 ⇒ Gestione portone garage

vediamoli uno per uno.

3.1. Server1

Questo server dovrà gestire 6 relè:

REL1 = Apertura tenda salotto
REL2 = Chiusura tenda salotto
REL3 = Apertura tenda cucina
REL4 = Chiusura tenda cucina
REL5 = Apertura e chiusura cancello carraio
REL6 = Accensione e spegnimento luci esterne

lo sketch dovrà tener conto anche del diverso funzionamento dei componenti:

Tenda del salotto ⇒ ha un funzionamento manuale, il pulsante deve essere premuto per tutto il tempo del movimento
Tenda cucina ⇒ ha un funzionamento automatico, basta premere il pulsante e rilasciare ma non si deve ripremere prima della fine corsa altrimenti lo sente come uno stop intermedio
Cancello carraio ⇒ Ha un funzionamento automatico, si preme lo stesso pulsante e si rilascia, sia per aprire che per chiudere
Luci esterne ⇒ Attualmente funzionano in manuale o in automatico (crepuscolare) mediante spostamento di un interruttore dopo averci pensato su parecchio ho deciso di fare uno sketch semplice che si occupi soltanto di attivare e disattivare i relè lasciando al software, lato web server, il compito di gestire i tempi, ho preso questa decisione in quanto è più facile modificare il software sul server via web che non modificare uno sketch e poi doverlo andare a caricare fisicamente sull'Arduino che sicuramente sarà inserito su delle scatole di derivazione e incastrato tra relè powerline e alimentatori.

Server1.ino

Modificare gli indirizzi IP con i vostri dati

```
// ========================================================
// = sketch per gestire 6 relays via internet             =
// =                                                      =
// = Autore: Walter62                                     =
// = Versione 1.00 del 06/08/15                           =
// ========================================================

#include <SPI.h>           // Libreria per comunicazione seriale
#include <Ethernet.h>      // Libreria per connessione internet
#include <EthernetUdp.h>   // Libreria UDP da: bjoern@cs.stanford.edu

// Inserire un indirizzo MAC e un indirizzo IP per la tua scheda.
// L'indirizzo IP dipende dalla tua rete locale:

byte mac[] = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x01 };
IPAddress ip(192, 168, 1, x);  // Inserire l'IP di Arduino
unsigned int localPort = 8888; // porta locale da mettere in ascolto

// buffers per ricevere e inviare dati

char packetBuffer[UDP_TX_PACKET_MAX_SIZE]; // buffer per contenere il pacchetto in
arrivo,
char ReplyBuffer[] = "ricevuto";           // stringa da ritornare come risposta

// Istanza EthernetUDP che ci permette di inviare e ricevere pacchetti via UDP

EthernetUDP Udp;

//const int LED = 7 assegno ad ogni relay un pin;

const int REL1 = 6; //primo relay
const int REL2 = 7; //secondo relay
const int REL3 = 8; //terzo relay
const int REL4 = 9; //quarto relay
const int REL5 = 4; //primo-2 relay
const int REL6 = 5; //secondo-2 relay

void setup() {

// avvio comunicazione Ethernet e UDP:

Ethernet.begin(mac,ip);
Udp.begin(localPort);

// imposto modalità dei pin pinMode(LED, OUTPUT):

pinMode(REL1, OUTPUT);
pinMode(REL2, OUTPUT);
pinMode(REL3, OUTPUT);
pinMode(REL4, OUTPUT);
pinMode(REL5, OUTPUT);
pinMode(REL6, OUTPUT);
```

```
// imposto lo stato dei pin digitalWrite(LED, HIGH-LOW);

digitalWrite(REL1, HIGH);
digitalWrite(REL2, HIGH);
digitalWrite(REL3, HIGH);
digitalWrite(REL4, HIGH);
digitalWrite(REL5, HIGH);
digitalWrite(REL6, HIGH);

Serial.begin(9600);
}

void loop() {

// se ci sono dati disponibili leggo un pacchetto

int packetSize = Udp.parsePacket();
if(packetSize){
   Serial.print("Received packet of size ");
   Serial.println(packetSize);
   Serial.print("From ");
   IPAddress remote = Udp.remoteIP();
   for (int i =0; i <4; i++)
     {
     Serial.print(remote[i], DEC);
     if (i <3)
        {
        Serial.print(".");
        }
     }
     Serial.print(", port ");
     Serial.println(Udp.remotePort());

// leggo il pacchetto all'interno del buffer

    Udp.read(packetBuffer,UDP_TX_PACKET_MAX_SIZE);
    Serial.print("Contents: ");
    Serial.println(packetBuffer);
    Serial.println(UDP_TX_PACKET_MAX_SIZE);
    if (String(packetBuffer) == "on1") {
      digitalWrite(REL1, LOW); //attivo il relay 1
    }
    if (String(packetBuffer) == "of1") {
      digitalWrite(REL1, HIGH); //disattivo il relay 1
    }
    if (String(packetBuffer) == "on2") {
      digitalWrite(REL2, LOW); //attivo il relay 2
    }
    if (String(packetBuffer) == "of2") {
      digitalWrite(REL2, HIGH); //disattivo il relay 2
    }
    if (String(packetBuffer) == "on3") {
      digitalWrite(REL3, LOW); //attivo il relay 3
    }
    if (String(packetBuffer) == "of3") {
      digitalWrite(REL3, HIGH); //disattivo il relay 3
```

```
    }
    if (String(packetBuffer) == "on4") {
      digitalWrite(REL4, LOW); //attivo il relay 4
    }
    if (String(packetBuffer) == "of4") {
      digitalWrite(REL4, HIGH); //disattivo il relay 4
    }
    if (String(packetBuffer) == "on5") {
      digitalWrite(REL5, LOW); //attivo il relay 5
    }
    if (String(packetBuffer) == "of5") {
      digitalWrite(REL5, HIGH); //disattivo il relay 5
    }
    if (String(packetBuffer) == "on6") {
      digitalWrite(REL6, LOW); //attivo il relay 6
    }
    if (String(packetBuffer) == "of6") {
      digitalWrite(REL6, HIGH); //disattivo il relay 6
    }

// inviamo una risposta, all'indirizzo IP e alla porta che ci hanno inviato il
pacchetto appena ricevuto

    Udp.beginPacket(Udp.remoteIP(), Udp.remotePort());
    Udp.write(ReplyBuffer);
    Udp.endPacket();
  }
delay(10);
}
```

Lo sketch da prima inizializza la scheda ethernet con un MAC e un IP fisso poi si mette in attesa di ricevere dei comandi via UDP, quando riceve un pacchetto lo filtra se c'è un comando valido lo confronta con le opzioni disponibili e se trova una corrispondenza attiva o disattiva il relè.

Se esegue un comando ritorna una risposta di comando eseguito al server che la aveva inoltrata.

3.2. Server2

Questo server dovrà gestire 2 relè:

REL5 = non usato
REL6 = Apertura e chiusura cancello carraio

Server2.ino

Modificare gli indirizzi IP con i vostri dati

```
//============================================================
//= sketch per gestire 6 relays via internet                 =
//=                                                          =
//= Autore: Walter62                                         =
//= Versione 1.00 del 06/08/15                               =
```

```
//=======================================================

#include <SPI.h>          // Libreria per comunicazione seriale
#include <Ethernet.h>     //  Libreria per connessione internet
#include <EthernetUdp.h>  // Libreria UDP da: bjoern@cs.stanford.edu

// Inserire un indirizzo MAC e un indirizzo IP per la tua scheda.
// L'indirizzo IP,DNS, gateway e subnet dipendono dalla tua rete locale:

byte mac[] = {0x00,0x00,0x00,0x00,0x00,0x01};
byte ip[]={192,168,1,x};         // Inserire l'IP del Arduino
byte DNS[]={192,168,1,254};
byte gateway[]={192,168,1,254};
byte subnet[]={255,255,255,0};
unsigned int localPort = 8888;  // porta locale da mettere in ascolto

// buffers per ricevere e inviare dati

char packetBuffer[UDP_TX_PACKET_MAX_SIZE]; //  buffer per contenere il pacchetto in
arrivo
char ReplyBuffer[] = "ricevuto";           // stringa da ritornare come risposta

// Istanza EthernetUDP che ci permette di inviare e ricevere pacchetti via UDP

EthernetUDP Udp;

// const int LED = 7 assegno ad ogni relay un pin;

const int REL1 = 6; //sk1 - primo relay
const int REL2 = 7; //sk1 - secondo relay
const int REL3 = 8; //sk1 - terzo relay
const int REL4 = 9; //sk1 - quarto relay
const int REL5 = 4; //sk2 - primo relay
const int REL6 = 5; //sk2 - secondo relay

void setup() {

// avvio comunicazione Ethernet e UDP:
if(Ethernet.begin(mac)==0) {
  Ethernet.begin(mac,ip,DNS,gateway,subnet); // Connessione con IP pre-impostato
}
delay(1000);           // Diamo alla Ethernet shield un secondo per inizializzarsi
Udp.begin(localPort);

// imposto modalità dei pin

pinMode(LED, OUTPUT):
//..
//..
// == segue come sketch precedente ==
```

Questo secondo sketch è uguale al primo salvo per la diversa inizializzazione della scheda che in questo caso è dinamica via DHCP, in effetti per questa applicazione non è l'ideale in quanto mi serve un IP fisso da raggiungere via software, però in molti router i device con IP fisso non vengono visualizzati e questo è uno dei modi per vedere se la scheda viene agganciata dal router, in questo caso però per avere sempre lo stesso IP bisogna agire sul router andando a impostare per l'IP della scheda il “lease infinito”.

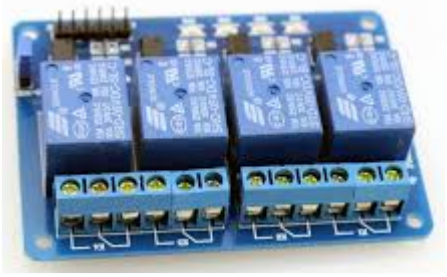
Io lo ho fatto come esercizio ma ciò non toglie che voi possiate mettere l'IP fisso su entrambi.

4. Collegamento dei relè su Arduino

Bene, ora che abbiamo i due server funzionanti andiamo ad eseguire i collegamenti fisici tra Arduino e i relè.

Per il server1 ho usato due schede relè, una da 4 e una da 2, mentre per il server2 ho usato una scheda da 2, ovviamente si può usare ciò che si ha a disposizione o che serve, di seguito le immagini delle due schede.





4.1. Collegamenti per Server1

Pin arduino	Sk1	Sk2
5V	Vcc	Vcc
GND	GND	GND
9	In1	
8	In2	
7	In3	
6	In4	
5		In1
4		In2

4.2. Collegamenti per Server2

Pin arduino	Sk2
5V	Vcc
GND	GND
5	In1
4	In2

5. Software di gestione

Per ovviare ai vari problemi di gestione dell'interfaccia grafica con i vari device ho riscritto l'intera interfaccia in JQuery mobile in modo che sia fruibile sia da Smartphone, dove è stata ottimizzata ma anche da tablet o da pc.

Trattandosi di webapp dovrebbe funzionare si qualsiasi terminale, io la ho testata su iOS , Crome e Firefox

Il software lo ho pensato modulare in modo che si possano implementare altre parti senza modificare l'esistente, in pratica ci sarà un programma per ogni componente da gestire, attualmente trovate i seguenti moduli:

1. Cancello
2. Tende
3. Luci
4. Termo

Il funzionamento del programma è abbastanza semplice:

Ad ogni pulsante viene abbinata una azione, quando il pulsante viene premuto viene inviata una istruzione al server Arduino abbinato il quale filtra l'istruzione, la confronta con quelle preimpostate e se trova una corrispondenza la esegue, terminata l'azione il sever Arduino risponde con un messaggio di avvenuta esecuzione e il software di gestione viene liberato e reso disponibile per un nuovo input

Alcuni pulsanti sono possono essere abbinati a più opzioni, è il caso del pulsante che apre contemporaneamente il cancello e il portone del garage che sono divisi su due server diversi, in questo caso bisogna far attenzione ad abbinarli correttamente sul codice e a controllare la sequenza di invio dei segnali ai server Arduino.

Per finire c'è il modulo “Termo” che non è un vero e proprio modulo, ma un link ad un sito esterno dove risiede il software vero e proprio per la gestione del sistema remoto.

Tutto il codice può essere scaricato dal mio portale nella sezione “Le mie guide” “Domotica open surce”

5.1. Installazione del software

Il software viene installato in locale, quindi è necessario poter raggiungere dall'esterno il web server per cui bisogna avere un IP pubblico fisso o nel caso lo si abbia dinamico appoggiarsi a dei servizi di DNS quali DtDns , NoIp o altri simili, in questo modo possiamo abbinare al nostro IP pubblico un nome di dominio facile da ricordare es. pippo.dtdns.net o domoticapippo.dtdns.net.

L'installazione del software è estremamente semplice basta andare sul raspberry, acquisire i diritti di amministratore e creare la directory “domotica” sulla radice del web server

```
..$ sudo -s
..# cd /var/www
..var/www# mkdir domotica
```

Scaricare il software dal mio sito Domotica

Con Filezilla copiare il contenuto della directory “domotica” all'interno della vostra directory “domotica”

Personalizzare i seguenti file inserendo le vostre impostazioni:

1. cancello.php ⇒ Inserire l'IP del server Arduino
2. tende.php ⇒ Inserire l'IP del server Arduino
3. luci.php ⇒ Inserire l'IP del server Arduino
4. header.php ⇒ per modificare o aggiungere intestazioni speciali relative agli smartphone
5. menu.php ⇒ per togliere e/o aggiungere voci e per impostare il corretto link ad siti esterni (es. per il pulsante “termo”)

Fatte le personalizzazioni lanciare il software digitando il percorso “ miosito/domotica/ ” si aprirà la pagina di login

dove dovete digitare la password (al primo avvio la password di default è “123456”), se usate il tastierino a video quando avete completato la password seguirà automaticamente, se invece usate la casella di input in alto dovrete cliccare su entra, una volta autenticati appare il menu principale

passare immediatamente al menu “Configura”

cambiare la password con una di vostro gradimento, adeguare il n° di caratteri alla lunghezza della password e se ritenete cambiate il titolo dell'applicazione, al termine cliccare sul pulsante “Salva dati”, questo è tutto.

Di seguito alcuni screenshot dei vari menu:

6. Test del sistema

A questo punto siamo pronti per testare il sistema e verificare che tutti i collegamenti siano corretti e il funzionamento sia come previsto.

Prima di collegare le utenze ai relè verificare che il tutto funzioni correttamente, lo potete fare sia da breadboard ma anche dal sistema già installato.

Da uno smartphone o PC aprite la pagina della gestione domotica

Ippubblico/domotica/

si apre la schermata di login dove dovete inserire la password o la homepage se la avete già inserita (di default è “123456” cambiatela immediatamente), provate a cliccare sui vari menu e di seguito sui pulsanti di controllo per verificare che il relativo relè si attivi (si sente il classico clik e si accende i relativo led sulla scheda relè) e nel caso di relè temporizzato che rimanga attivato per il tempo necessario.

Fate il controllo su tutti i menu e pulsanti, anche quelli che non intendete usare per assicurarvi che non ci siano dei comportamenti anomali, se tutto funziona regolarmente eseguite un ultima prova, togliete alimentazione agli Arduino e dopo un minuto ridate alimentazione per simulare un blackout elettrico, alla riaccensione verificate che nessun relè si attivi senza che voi impartiate il comando.

ATTENZIONE: è molto importante che il sistema venga testato bene prima di usarlo sull'impianto di casa per evitare che eventuali comportamenti anomali danneggino gli apparati o facciano danni.

7. Integrazione dei server sull'impianto

A questo punto ci troviamo con un mucchio di componenti sopra al tavolo collegati con dei fili tra di loro, con gli alimentatori e con la rete domestica da connettere, neccessita un po di ordine....

In primo luogo dobbiamo mettere tutti i componenti che fanno parte di un server all'interno di una scatola per poi poterla installare a parete nella posizione più opportuna, dobbiamo anche risolvere il problema del cavo LAN, se avete la possibilità di stendere un cavo meglio altrimenti dobbiamo ricorre alle power line



ne serve una sul router e una per ogni server Arduino, nel nostro caso ne servono 3, quindi su ogni scatola dobbiamo mettere:

- n° 1 Arduino + scheda ethernet
- n° 1 alimentatore (5V se si usa la USB o 7÷12V se si usa il jeck) io opto per 9V
- n° 1 powerline
- n° 1 o 2 schede rele
- n°1 ciabatta per collegare alimentatore e powerline

è un bel po di roba se si considera che poi ci saranno i cavi per l'alimentazione, il cavo di rete tra powerline e Arduino, il cavo dell'alimentatore e i cavi per la connessione dei relè.

Ovviamente l'obbiettivo è di usare una scatola il più piccola possibile, io con schede relè a 4 posti ho usato scatole di derivazione da 190x170x70 per schede relè più grandi bisogna passare a scatole maggiori.

Di seguito alcune foto che mostrano come ho assemblato il tutto:

**ATTENZIONE: ** qualsiasi operazione fatta sull'impianto elettrico può provocare gravi danni a cose e persone, tutte le operazioni descritte in seguito devono essere eseguite da personale esperto e qualificato, non ci riteniamo responsabili per eventuali danni a cose e/o persone

sul fondo ho fissato 4 colonne per poter fissare una piastra che supporterà l'elettronica mentre sotto passeranno tutti i fili

a scatola fissata al muro con i cavi già passati e la ciabatta già collegata.Per la ciabatta vi consiglio quelle non cablate e la più economica possibile, senza interruttori o altro, i cavi di alimentazione devono arrivare direttamente alle piastre di contatto con le spine, questo per ridurre al massimo le interferenza sulle powerline.

Connessione powerline, alimentatore e disposizione cavi

Fissaggio della piastra per supporto elettronica

Risultato finale con tutto collegato

Dopo aver fatto tutto questo lavoro posso affermare che se qualcuno sta ristrutturando casa o la sta costruendo nuova consiglio vivamente, almeno di predisporre le canaline (belle grandi) per portare la rete ethernet e l'alimentazione 5V dc su ogni stanza e di predisporre delle scatole belle grandi, alla fine si risparmino soldi e tempo e il tutto rimane invisibile.

A questo punto abbiamo montato tutto e possiamo passare a collegare fisicamente il nostro sistema alle utenze, dato che l'impianto è già fatto ho cercato di fare una cosa il meno invasiva possibile pertanto ho deciso di utilizzare i pulsanti già esistenti per interfacciarmi con il sistema domotico, ovvero ho messo i relè in parallelo ai pulsanti esistenti, a questo scopo le scatole contenenti i server con i relè devono essere messe in posizioni strategiche da dove sia possibile raggiungere i pulsanti o dei punti di connessione con essi, a voi scegliere la soluzione migliore, magari fatevi aiutare da un elettricista a scegliere il luogo adatto, di seguito lo schema di connessione ai relè.

Server	**Scheda rele**	**Rele**	**Utenza**
Server1	Scheda 4R	R1	Apertura tenda salotto (attivo per tutto il tempo di apertura)
		R2	Chiusura tenda salotto (attivo per tutto il tempo di apertura)
		R3	Apertura tenda cucina (attivazione ad impulso)
		R4	Chiusura tenda cucina (attivazione ad impulso)
	Scheda 2R	R1	Apertura e chiusura cancello carraio (attivazione ad impulso)
		R2	Accensione luci esterne (attivo per tutto il tempo di apertura)
Server2	Scheda 2R	R1	= Libero =
		R2	Apertura e chiusura portone garage (attivazione ad impulso)

8. Costi

Di seguito un riepilogo dei costi a solo scopo informativo

Componente	n° Pz	€/Pz	Costo €
Raspberry Pi B+	1	26	26
SD card 4 GB	1	8	8
Case	1	6	6
Arduino (CCDuino)	2	6	12
Ethernet WZ5100	2	8.50	17
Scheda rele x4	1	8	8
Scheda rele x2	2	4	8
Alimentatore 5V 1A	2	11	22
Scatole di derivazione	2	8	16
Ciabatte multipresa	2	8.5	17
TP Link	3	12	36
TOTALE			176

Quando costruii l'impiato un anno fa non esistevano ancora ma se al posto di Arduino + WA5100 + TP Link al costo di 26.50€ si usa una Wemos D1 mini che integra l'WIFI al costo di 3.5€ si possono risparmiare 46 €

9. Licenza d'uso

Copyright © 2012 - 2016 by Walter62. All rights reserved

Quest'opera è distribuita con licenza:

Creative Commons Attribuzione - Non commerciale - Condividi allo stesso modo 3.0 Italia.

Per leggere una copia della licenza visita il sito web:

http://creativecommons.org/licenses/by-nc-sa/3.0/it/

o spedisci una lettera a:

Creative Commons
171 Second Street, Suite 300
San Francisco, California, 94105, USA.

Fonte:
Wiki - Progetti

Autore:
Walter62

Ultimo aggiornamento: **2023/01/01 17:03**