Documentazione SW Sedute Procida

Sommario

Descrizione del software

Tramite un access due RaspBerry Pi comunicano come client e server. Lo scopo dell'software è mandare gli input del RaspBerry a cui sono collegate le sedute ad un altro RaspBerry che monta un WebServer che dà in output sui pin GPIO un segnale, che attiva un relè, in modo da accendere un qualsiasi dispositivo che necessiti della corrente a 220V.

Il linguaggio utilizzato per la realizzazione del software è Python 3.

Importante sapere che in Python non bisogna dichiarare il tipo delle variabili, che viene autodeterminato, e non esistono parentesi graffe, è tutto gestito con le indentazioni.

Librerie utilizzate

- RPi.GPIO (Gestione dei Pin-On-Board)[1]
- os (Libreria che permette di eseguire una serie di funzioni del sistema operativo)[2]
- http.server (Libreria per utilizzare il protocollo http)[3]

Descrizione del WebServer (Server)

Il WebServer ha lo scopo di ricevere dei comandi tramite un CURL[4](Client Uniform Resource Locator) nel nostro caso avvia un file nella cartella \public\cgi-bin a seconda dell'input ricevuto dalla seduta.

File System

- wirelessButton (Main Root)
    - webs.py (File da eseguire per aprire il WebServer)
    - public (cartella resa pubblica dal WebServer)
        - cgi-bin (cartella che contiene i file eseguiti dai CURL del client)
            - spengo1
            - spengo2
            - spengo3
            - accendo1
            - accendo2
            - accendo3

Descrizione dei singoli file

webs.py

Il file utilizza le librerie os e http.server, da quest'ultima richiama solamente le funzioni:

- HTTPServer che permette l'utilizzo del protocollo http e di aprire effettivamente il server.
- CGIHTTPRequestHandler che permette al client di visualizzare e gestire l'avvio dei file sul server.[5]

Dichiarazioni variabili e funzioni

- La funzione CGIHTTPRequestHandler viene dichiarata come handler
- La funzione HTTPServer è dichiarata come server_object

Note al Codice

- Alla riga 6 viene dichiarata la cartella con le risorse per la CGI utilizzando il metodo cgi_directories della funzione CGIHTTPRequestHandler

- Alla riga 7 viene dichiarata la cartella su cui il server deve cercare le varie risorse tramite il metodo chdir della libreria os.
- Alla riga 9 viene dichiarato l'indirizzo del server su '', che equivale a dire che il server è sull'indirizzo locale (localhost), la porta, che abbiamo impostato su 9000 e viene dichiarato che il server non è statico ma dinamico utilizzando la CGI[6] il tutto è stato dichiarato nella funzione HTTPServer
- Alla riga 11 tramite il metodo serve_forever della libreria HTTPServer il WebServer viene eseguito.

accendo1.py/ accendo2.py/ accendo3.py

Questi 3 file funzionano nello stesso identico modo, l'unica cosa che varia è il pin GPIO sul RaspBerry a cui sono collegati. Vista la differenza nulla nel codice, sarà documentata in modo "neutro".

Tali file vengono richiamati dai curl ricevuti dal WebServer e contengono il codice che permette l'accensione dei dispositivi collegati al circuito.

Dichiarazione variabili e funzioni

- La libreria RPi.GPIO è dichiarata come GPIO.
- I pin utilizzati sono il 12,18, 22

Note al codice

- Alla riga 4 viene dichiarato che i pin verranno gestiti seguendo la numerazione sulla scheda, per farlo abbiamo utilizzato il metodo setmode della funzione RPi.GPIO, impostando il parametro sul "GPIO.BOARD"
- Alla riga 5 tramite il metodo setwarnings della funzione RPi.GPIO impostiamo che il codice non deve restituire messaggi di attenzione quando viene eseguito (potrebbero causare un blocco nel programma)
- Alla riga 6 impostiamo tramite il metodo setup della funzione RPi.GPIO il pin su cui come scritto nel campo successivo verrà eseguito l'output utilizzando il parametro GPIO.OUT[7]
- Riga 7 tramite il metodo output della funzione RPi.GPIO diciamo che sul pin inizializzato nel setup deve esserci un output GPIO.HIGH, il che significa che sul pin fisicamente passerà una corrente di 5V finché non verrà modificata.

spengo1.py/ spengo2.py/ spengo3.py

Questi 3 file funzionano nello stesso identico modo, l'unica cosa che varia è il pin GPIO sul RaspBerry a cui sono collegati. Vista la differenza nulla nel codice, sarà documentata in modo "neutro".

Tali file vengono richiamati dai curl ricevuti dal WebServer e contengono il codice che permette lo spegnimento dei dispositivi collegati al circuito.

Dichiarazione variabili e funzioni

- La libreria RPi.GPIO è dichiarata come GPIO.
- I pin utilizzati sono il 12,18, 22

Note al codice

- A riga 4 viene dichiarato che i pin verranno gestiti seguendo la numerazione sulla scheda, per farlo abbiamo utilizzato il metodo setmode della funzione RPi.GPIO, impostando il parametro sul "GPIO.BOARD"
- A riga 5 tramite il metodo setwarnings della funzione RPi.GPIO impostiamo che il codice non deve restituire messaggi di attenzione quando viene eseguito(potrebbero causare un blocco nel programma)

- Alla riga 6 impostiamo tramite il metodo setup della funzione RPi.GPIO il pin su cui come scritto nel campo successivo verrà eseguito l'output utilizzando il parametro GPIO.OUT
- A riga 7 tramite il metodo output della funzione RPi.GPIO diciamo che sul pin inizializzato nel setup deve esserci un output GPIO.LOW, il che significa che sul pin fisicamente non passerà alcuna corrente.
- A riga 8 tramite il metodo clenup della funzione RPi.GPIO puliamo completamente il pin come se non fosse stato inizializzato.

Descrizione del Client

Il client ha lo scopo di mandare al server il segnale che ogni seduta manda ad uno dei pin di controllo sul RaspBerry. Il programma è racchiuso in un solo file.

Descrizione del file workingButton.py

Il programma usa le librerie os, e RPi.GPIO, ed utilizza gli if[8] per determinare quale curl mandare a seconda del pin che viene utilizzato (es. Se il pin 12 che è il primo pin riceve corrente dalla seduta abbassata verrà mandato un curl che aprirà il file accendo1.py sul WebServer accendendo il dispositivo collegato al relè collegato al pin 12 del WebServer, nel momento in cui la seduta verrà alzata manderà un curl che aprirà il file spengo1.py, che spegnerà il dispositivo prima citato). Nel file è presente anche un ciclo while[9], questo ciclo è infinito, permettendo il controllo continuo dello stato dei pin. Tutto quello che non viene documentato in questo momento è perché ad un'osservazione attenta solo presenti variabili e if che hanno un impatto nullo sul funzionamento del programma e sono stati utilizzati solo per test preliminari sul codice.

Dichiarazione variabili e funzioni

- La funzione RPi.GPIO viene dichiarata come GPIO
- Le variabili presenti nel programma sono:
    - ip: variabile in cui viene inserito l'ip del WebServer
- I pin utilizzati sono il 12, 18 e 22.

Note al codice

- A riga 5 viene dichiarato che i pin verranno gestiti seguendo la numerazione sulla scheda, per farlo abbiamo utilizzato il metodo setmode della funzione RPi.GPIO, impostando il parametro sul "GPIO.BOARD"
- Alle righe 8, 11 e 14 impostiamo tramite il metodo setup della funzione RPi.GPIO i pin su cui come scritto nel campo successivo verranno letti gli input utilizzando il parametro GPIO.IN
- A riga 28 inizia il loop del programma.
- L'if a riga 30 è uguale per funzionamento a quelli di riga 46 e 62, il numero del pin ed il file a cui fa il curl, quindi verrà preso in esame in solo quello di riga 30 e verranno segnate le varianti presenti negli if di riga 46 e 62. Questo if verifica lo stato del pin 18, se il pin riceve corrente e quindi il suo parametro sarà GPIO.HIGH eseguirà i comandi al suo interno
    - A riga 46 il pin controllato è il 22
    - A riga 62 il pin controllato è il 12
- A riga 32 tramite il metodo system della libreria os inviamo una richiesta curl in cui viene aperto il file accendo1.py[10]
    - A riga 48 il file aperto tramite curl è accendo2.py
    - A riga 64 il file aperto tramite curl è accendo3.py
- L'if a riga 39 è uguale per funzionamento a quelli di riga 55 e 71, il numero del pin ed il file a cui fa il curl, quindi verrà preso in esame in solo quello di riga 39 e verranno segnate le varianti presenti

negli if di riga 55 e 71. Questo if verifica lo stato del pin 18, se il pin non riceve corrente e quindi il suo parametro sarà GPIO.LOW eseguirà i comandi al suo interno
    - A riga 55 il pin controllato è il 18
    - A riga 71 il pin controllato è il 22
- A riga 41 tramite il metodo system della libreria os inviamo una richiesta curl in cui viene aperto il file spengo1.py
    - A riga 57 il file aperto tramite curl è spengo2.py
    - A riga 73 il file aperto tramite curl è spengo3.py

Note Tecniche

I vari programmi sono stati impostati sui RaspBerry in autorun, impostando l'avvio automatico nel file rc.local del RaspBerry.[11] Rc.local è un file presente in tutte le distribuzioni linux ed è possibile impostare al suo interno degli script da eseguire all'avvio del dispositivo.

L'istruzione utilizzata per mettere in autorun il WebServer è:

su pi -c '/usr/bin/python3 /home/pi/Desktop/wirelessButton/webs.py &'

Solitamente le istruzioni in rc.local vengono eseguiti come se fossero lanciate da un amministratore (come se venisse usato il sudo), il WebServer per questioni di sicurezza restituisce un errore quando viene avviato come amministratore, quindi tramite il permessi di su (super user) facciamo eseguire il comando all'utente base pi, la versione generalizzata del comando è:

su "Nome Utente del sistema" -c ' "Istruzione da eseguire" &'

Per quanto riguarda il programma nel client è bastato scrivere il comando per l'avvio del file:

/usr/bin/python3 /home/pi/Desktop/wirelessButton/workingButton.py

Sul router sono stati impostati gli ip statici[12] per entrambi i RaspBerry:

- 192.168.1.83 è l'indirizzo del WebServer
- 192.168.1.135 è l'indirizzo del client

Per garantire la possibilità di eseguire una manutenzione, sono abilitati sia la connessione ssh che vnc.[13]

Note

1. Documentazione RPi.GPIO (https://pythonhosted.org/RPIO/)
2. Documentazione os (https://docs.python.org/3/library/os.html)
3. Documentazione http.server (https://docs.python.org/3/library/http.server.html)
    a. Un protocollo gestisce cosa fa una determinata macchina in una rete, il protocollo http permette a chi si connette di visualizzare pagine web, da cui deriva il nome HTTP (HyperText Transfer Protocol)
4. i CURL similmente agli URL utilizzati per la navigazione Internet, indicano la posizione specifica di una risorsa specifica nel WebServer
5. CGI (Common Gateway Interface), è un'integrazione del protocollo http, che permette ai client non solo di visualizzare staticamente il WebServer, ma dà la possibilità di interagirci dinamicamente (un

esempio è che su internet non puoi avviare un file sul server di google, mentre è proprio il funzionamento base del nostro programma).

6. L'indirizzo è uno dei vari identificativi univoci di un dispositivo sulla rete, la porta serve a gestire il trasporto dei dati in una rete (varie richieste vanno gestite su porte diverse per non intasare la rete, che ne subirebbe rallentata.)
7. Questa riga funziona in modo simile alla dichiarazione di una variabile, di fatto inizializziamo il pin e gli diciamo che deve essere pronto a restituire un output.
8. L'if è utilizzato per indicare al programma cosa fare in certe condizioni.
9. Il while è un ciclo che ripete le stesse istruzioni nel momento in cui una condizione viene soddisfatta, quando quella condizione viene a mancare il loop si interrompe, nel nostro caso la condizione è sempre vera e quindi il programma è in loop continuo
10. "curl http://" + ip + ":9000/cgi-bin/accendo1.py" dove:
    a. Curl è il tipo di richesta
    b. http:// è il protocollo che utilizziamo per interagire con il WebServer
    c. ip è la variabile contenente l'ip del server
    d. :9000 è la porta in ascolto del server
    e. /cgi-bin/ è il percorso del file a partire dalla cartella pubblica che abbiamo dichiarato nel codice del WebServer
    f. accendo1.py è il programma da eseguire
11. /etc/rc.local
12. Gli ip come standard sui router sono dinamici, cambiano ad ogni tentativo di connessione alla rete, ciò implica che un dispositivo non sempre potrebbe avere lo stesso indirizzo ip, e quindi non poter raggiungere l'indirizzo indicato nella variabile ip del programma workingButton.py nel client.
13. L'SSH (Secure Shell) è un protocollo per stabilire una connessione sicura tra due dispositivi, questa però è limitata alla sola riga di comando, quindi l'utente dovrà navigare nel dispositivo utilizzando i comandi. Il VNC invece è un software che permette di connettersi da remoto su un dispositivo utilizzando la sua interfaccia grafica, quindi in pratica visualizzare da remoto il desktop del RaspBerry.