



Università degli Studi di Napoli Federico II

SCUOLA POLITECNICA E DELLE SCIENZE DI BASE

Corso di Laurea in Ingegneria Informatica

PROGETTO DI TECNOLOGIE INFORMATICHE PER L'AUTOMAZIONE INDUSTRIALE

Terraforming Mars

Candidati:

Marco Esposito N46002483

Giorgio di Costanzo N46003564

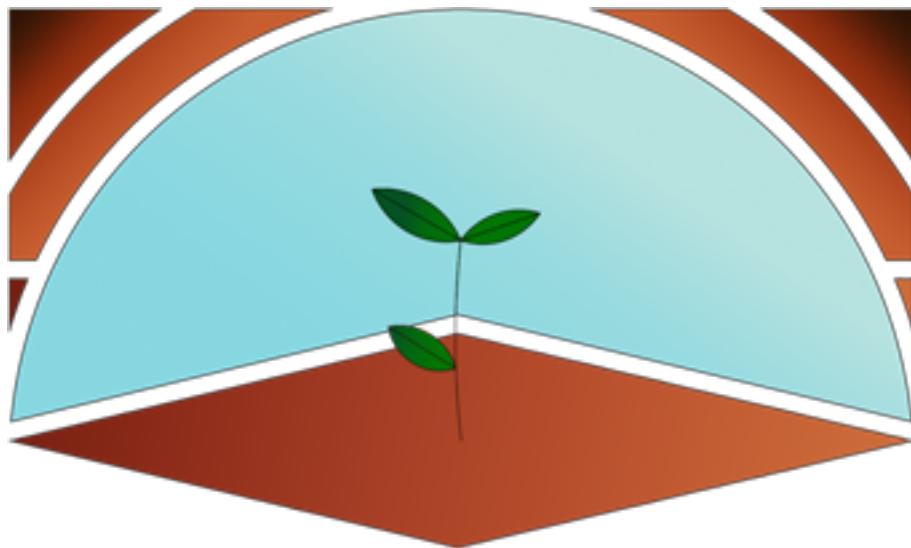
Davide De Martino N46003374

Docente:

Prof. Adriano Mele

Capitolo 1

Introduzione



L'uomo è agli albori di una nuova fase esplorativa. I recenti traguardi in campo spaziale hanno ispirato la realizzazione di questo progetto che si pone come obiettivo quello di presentare una possibile soluzione di terraformazione del pianeta Marte. L'obiettivo principale è quello di creare un'atmosfera abitabile, poiché le condizioni di vita sul pianeta rosso sarebbero impossibili a causa di diversi fattori: la forte escursione termica, che vede temperatura passare dai -150C° ai 20C°, e la presenza di un'atmosfera rarefatta, incapace di trattenere al di fuori del pianeta radiazioni solari incompatibili per la vita. Il modo in cui il modulo di terraformazione pensato in questo progetto, si propone di risolvere questo problema è quello di avviare un processo di automazione che produca ossigeno da immettere nell'atmosfera di Marte. Gli atomi di ossigeno, stimolati dalle forti radiazioni ultraviolette che colpiscono Marte, tendono a stabilire dei legami tra di loro: tre atomi di ossigeno costituiscono una molecola di ozono. Sulla Terra l'ozono è la principale barriera che protegge la vita dalle radiazioni solari.

Si è pensato ad un sistema automatico a forma di cupola per la coltivazione di vegetali, in cui simulare un'atmosfera terrestre, replicando la pressione atmosferica e le temperature tipiche di un clima mite. La cupola si occuperà di mantenere favorevoli le condizioni climatiche attraverso una serie di routine automatizzate a cui si affiancherà il processo di estrazione dell'ossigeno. Queste routine prevedono anche un sistema di emergenza della cupola, considerate le forti tempeste di sabbia del pianeta che potrebbero oscurare il passaggio della luce all'interno. Lo strato esterno della cupola avrà la funzione che svolge l'ozono sulla Terra, ovvero schermare l'interno della cupola dalle radiazioni. Si è pensato quindi ad una soluzione già impiegata nello spazio per ovviare a questo problema: il Lexan. Il Lexan

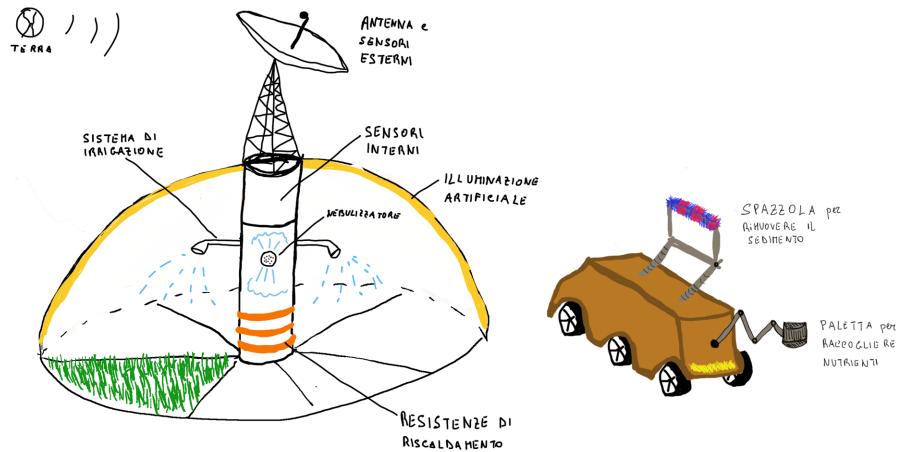


Figura 1.1: una prima bozza della cupola e del rover

è una particolare resina appartenente alla famiglia dei policarbonati, che ha la caratteristica di essere opaca alle radiazioni solari ultraviolette e per questo già impiegata in passato nelle visiere dei caschi delle tute spaziali durante le missioni lunari.

La cupola otterrà l'energia necessaria al suo funzionamento mediante l'uso di pannelli solari. I pannelli chiusi fungeranno da guscio per la cupola durante il viaggio, mentre al momento dell'atterraggio e dopo il segnale di start dalla terra, questi si dispiegheranno iniziando a fornire energia, donandole al contempo un'evocativa forma a fiore.

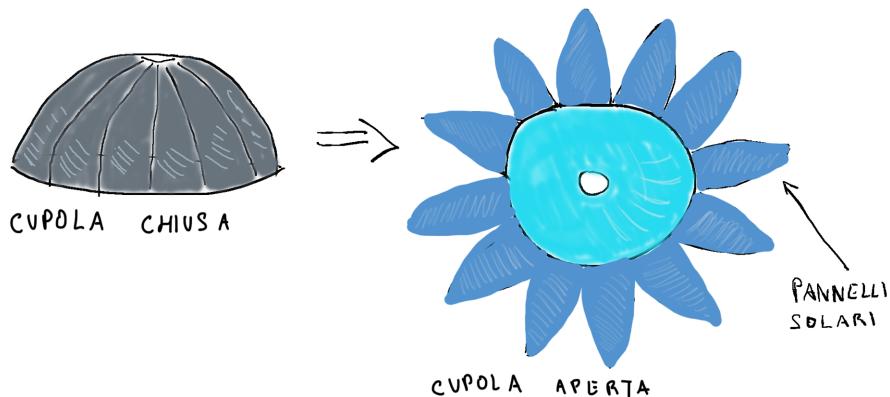


Figura 1.2: bozza dei pannelli solari

Per la realizzazione di questo modulo di terraformazione ci serviremo degli strumenti acquisiti durante il corso di tecnologie informatiche per l'automazione industriale del prof. Adriano Mele. Tali strumenti prevedono l'uso dei linguaggi di programmazione SFC e LADDER che permettono un automatismo robusto. A questi si affiancherà un'interfaccia HMI (Human machine interface) che permetterà ad un team da Terra di monitorare lo stato della cupola, ed un controllore PID che si occuperà della gestione della temperatura all'interno di essa. L'ambiente di sviluppo proposto per la realizzazione dei programmi SFC e LADDER è OpenPLC Editor. I programmi gireranno sul motore OpenPLC Runtime che comunicherà con l'HMI ScadaBR. L'implementazione del PID avrà per mezzo del software Matlab/Simulink.



Capitolo 2

Programmi SFC

Un segnale di atterraggio ed uno start dalla Terra inizializzeranno il processo di automazione della cupola. Il guscio di pannelli solari verrà dischiuso e la cupola inizierà a ricevere energia. La temperatura verrà portata inizialmente a 10C° per favorire la gestione dei flussi d'acqua nelle successive fasi. All'interno della cupola, raggiunta tale temperatura, avverrà l'immissione dei principali gas che costituiscono l'atmosfera terrestre: Ossigeno, Azoto ed Argon. Si è scelto per semplicità di trattare solo questi tre gas in quanto sono quelli presenti in maggior parte nell'atmosfera Terrestre. Tuttavia questo non esclude la necessaria presenza del resto dei gas terrestri per un corretto svolgimento della vita nell'ambiente simulato. Parallelamente sarà regolata l'umidità ambientale portandola ad un 50% di umidità relativa per mezzo di umidificatori ad ultrasuoni, e verrà avviata un'irrigazione preparatoria del terriccio compresso e liofilizzato. Trascorso il tempo necessario a raggiungere i valori desiderati, si prosegue con una fase di semina differenziata: la prima metà della superficie sarà seminata con Aloe Vera, mentre la restante parte sarà seminata con Edera Infestante. La scelta è ricaduta su queste due varietà considerata la resistenza a condizioni climatiche avverse e la naturale predisposizione alla produzione di ossigeno e alla cattura di monossido di carbonio ed altri agenti inquinanti. Una volta conclusa la fase di semina, l'SFC di preparazione, darà un segnale di "start" a quelle che saranno le routine del nostro sistema di automazione.

I tempi nei seguenti programmi SFC sono stati scelti in funzione di una simulazione e quindi non corrispondono ai reali tempi necessari a svolgere le azioni previste.

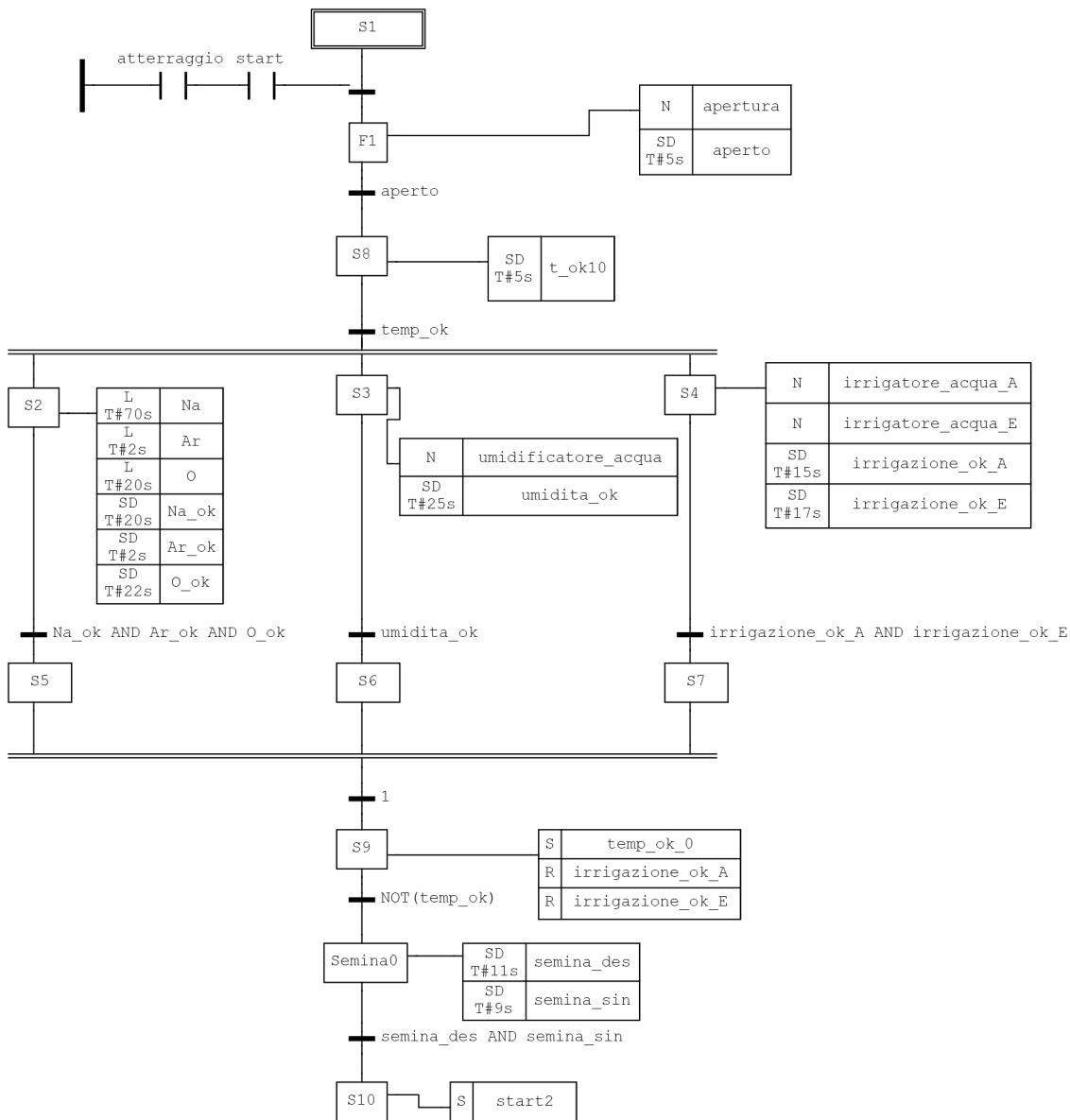


Figura 2.1: SFC di inizializzazione



Le routine saranno regolate in base ad un ciclo giorno/notte che simulerà l'alternarsi delle fasi solari. In una reale implementazione di questo modulo di terraformazione questa informazione sarà calcolata in base alla posizione della cupola sul pianeta e alla rivoluzione di quest'ultimo.

Nel nostro progetto il giorno e la notte si susseguiranno ogni 90 secondi.

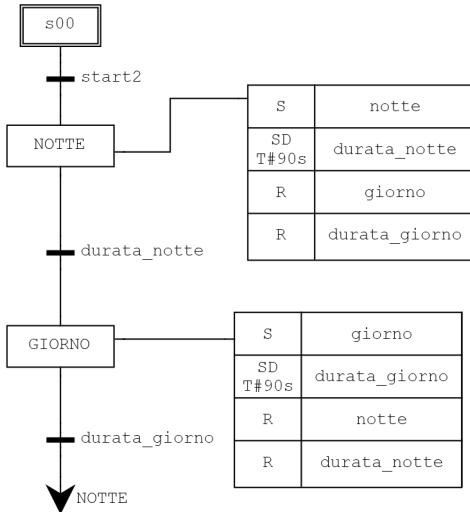


Figura 2.2: SFC per la regolazione del ciclo Giorno/Notte

Le azioni che il nostro programma svolgerà si differenzieranno in base al giorno e alla notte. Di notte verrà attivata una ventilazione parziale e sarà regolata la temperatura sui 18C°. Queste condizioni si manterranno fino al sorgere del Sole. Successivamente, di giorno, sarà attivata una ventilazione completa e la temperatura verrà portata a 24C°. L'emissione di ossigeno, da parte di un vegetale, avviene durante il giorno e pertanto in questa fase si susseguiranno operazioni di prelievo di ossigeno dalla cupola ed espulsione di quest'ultimo al di fuori di essa. Come precedentemente anticipato, è prevista la gestione di un'emergenza. Se durante il giorno, il sensore di luminosità, all'interno della cupola, percepisce un insufficiente afflusso di luce, lancerà un'emergenza gestita per mezzo di un Rover. Lo stato della cupola verrà settato su "sporco" e verrà poi attivato un impianto di illuminazione ausiliario. Nel caso in cui finisca il giorno prima della risoluzione di questa emergenza, l'impianto di illuminazione ausiliario verrà disattivato. Se lo stato della cupola verrà ripristinato prima della fine del giorno, il segnale di emergenza, dato dal sensore di luminosità, verrà resettato così come le luci ausiliarie. La conclusione di ogni ramo di questo SFC rimanderà al suo stato di quiete. Il ramo "giorno" dell'SFC sarà eseguito anche durante la situazione di emergenza.



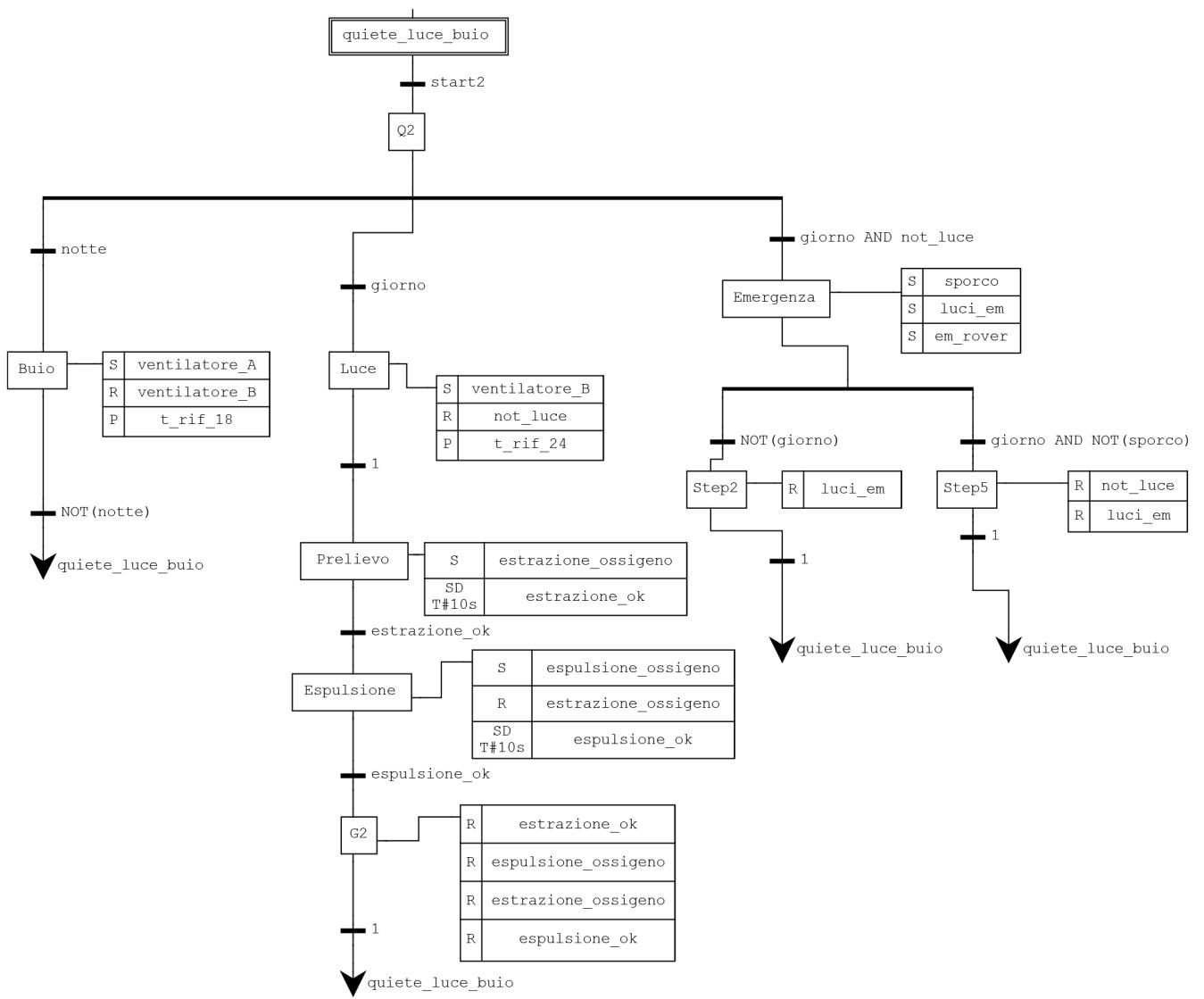


Figura 2.3: SFC delle routine Giorno/Notte



Il segnale "em_rover" sul fronte di salita farà scattare la routine di emergenza. I segnali di semina dovranno essere attivi per essere certi che i rover non saranno impegnati durante tale fase. A questo punto, il rover si staccherà dal suo dock di ricarica per portarsi fino al perimetro della cupola (sensore_P). Raggiunta la cupola, verrà estratta una spazzola che entrerà in funzione. Durante tale funzionamento, il rover percorrerà il perimetro della cupola per il tempo necessario ad assicurarne la completa pulizia. Conclusasi questa fase, il rover verrà arrestato e lo stato della cupola verrà ripristinato su "pulito". Quindi, quest'ultimo, percorrerà a ritroso il percorso fino al suo dock di ricarica. Infine, lo stato di emergenza del rover verrà ripristinato e l'SFC si posizionerà in attesa di una nuova emergenza.

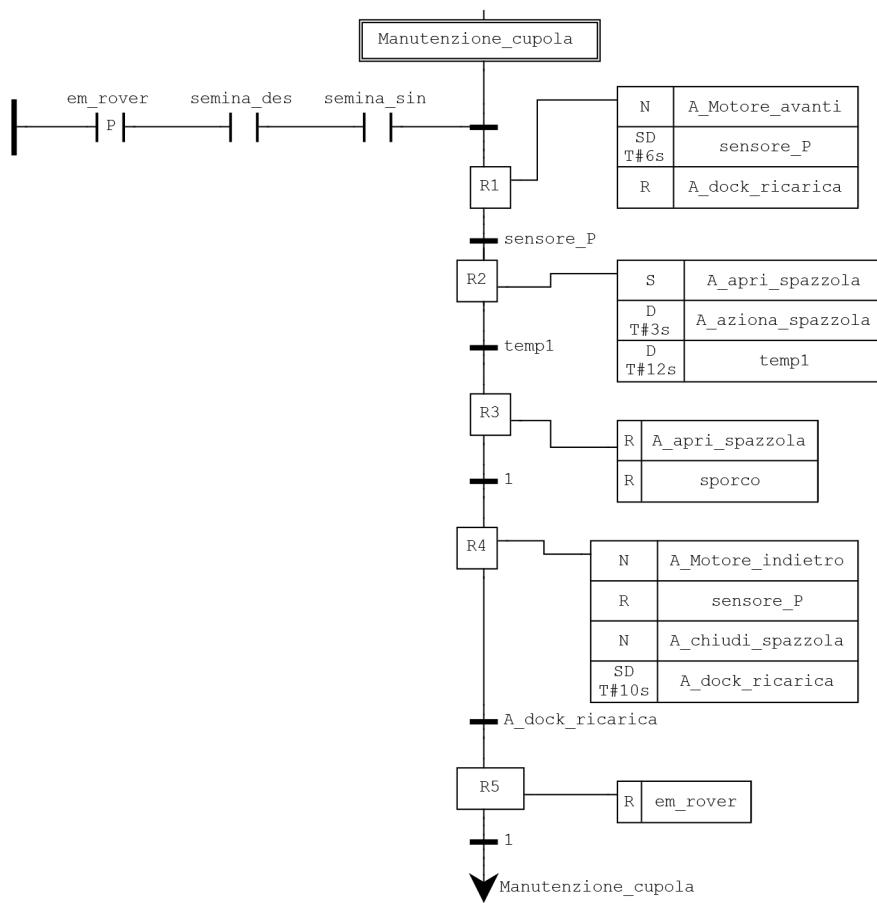


Figura 2.4: SFC dell'emergenza Rover



Indipendentemente da tutte le azione compiute dai nostri SFC, sarà effettuata un'irrigazione automatica indipendente a seconda del tipo di pianta, all'attivarsi di un segnale "secco" dato da un sensore di umidità presente nel terreno. Ancora una volta abbiamo simulato l'attivazione di "secco" per mezzo di un timer allo scopo di simulare il funzionamento del sensore. L'aloë vera, essendo una pianta secca riceverà un'irrigazione meno frequente rispetto all'edera.

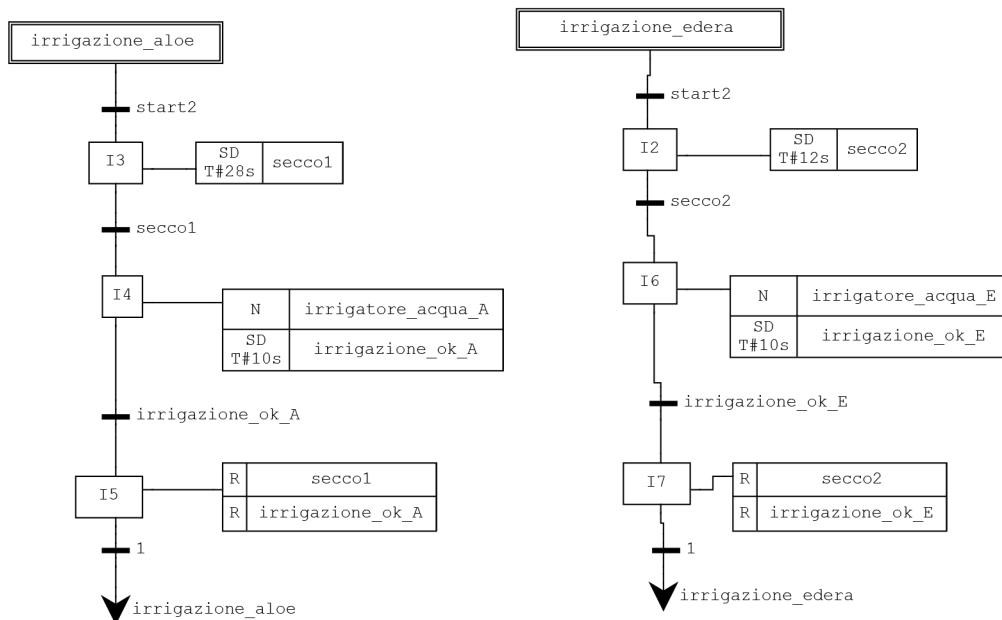


Figura 2.5: SFC di irrigazione



2.1 Semina

Per quanto riguarda la semina, è valsa la pena di implementare un SFC più dettagliato. Si è scelto di operare per mezzo di due rover a cui è richiesto di attraversare un tunnel che porta alla cupola in mutua esclusione. Nel tunnel avviene una pressurizzazione in fase di andata ed una depressurizzazione in fase di ritorno. Il tunnel viene quindi gestito come risorsa esclusiva per mezzo di un costrutto semaforo. Completate le fasi di inizializzazione di atmosfera, umidità e irrigazione, i due rover ricevono lo start per avviare la semina. Al segnale di start il rover A si staccherà dal suo dock di ricarica azionando il motore in avanti, portandosi in una zona di attesa. Esso transiterà nella fase successiva se e solo se riceverà l'ok dal semaforo. Una volta imboccato il tunnel, il semaforo revokerà il permesso di transitare al rover B. Il rover A si muoverà in avanti fino a raggiungere un sensore "C" posto all'ingresso del tunnel. A questo punto avverrà la fase di depressurizzazione che porterà la pressione del tunnel al livello della pressione sul pianeta. Si potrà quindi aprire il portello A del tunnel che si richiuderà una volta che il rover avrà raggiunto un secondo sensore "D" all'interno del tunnel. Il rover continuerà ad andare avanti fino a raggiungere un sensore "E" posto in prossimità del portello di entrata nella cupola. In questa fase avverrà la pressurizzazione del tunnel, dove la pressione verrà portata ai livelli di quella della cupola. Il portello B quindi si aprirà lasciando transitare il rover all'interno di essa dirigendosi verso la sua zona di semina (g_sx_sensor). Una volta che il rispettivo rover avrà raggiunto la sua zona di semina, la risorsa tunnel verrà liberata per lasciarsi attraversare dall'altro rover che si troverà in zona di attesa. Per un tempo congruo, il rover rilascerà sul suo percorso i semi da piantare. Una volta concluso il suo compito, il rover dovrà percorrere a ritroso il tragitto fino a ripetere l'attraversamento del tunnel al contrario per poi riposizionarsi nel dock di ricarica. A questo punto verrà inviato un segnale dell'avvenuta semina. Il tragitto percorso dal rover è frutto di un'intelligenza artificiale. Per praticità si è deciso di non includere l'SFC della semina nell'SFC principale.

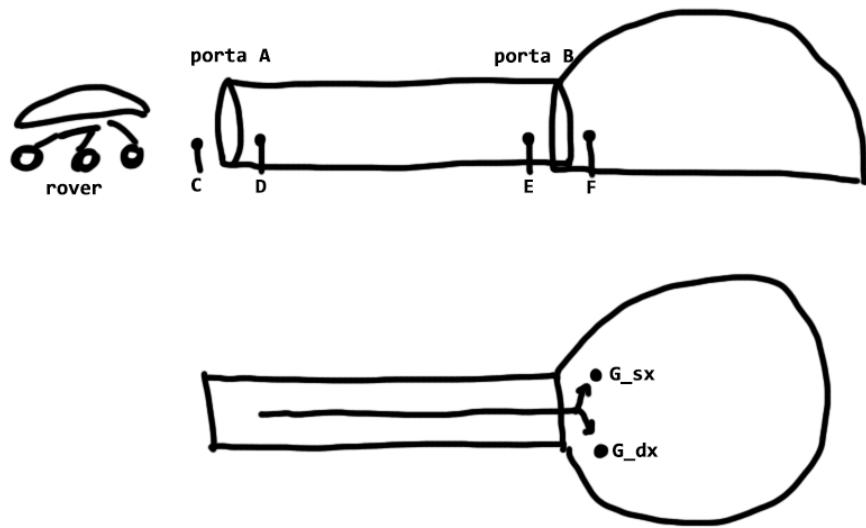


Figura 2.6: bozza della semina



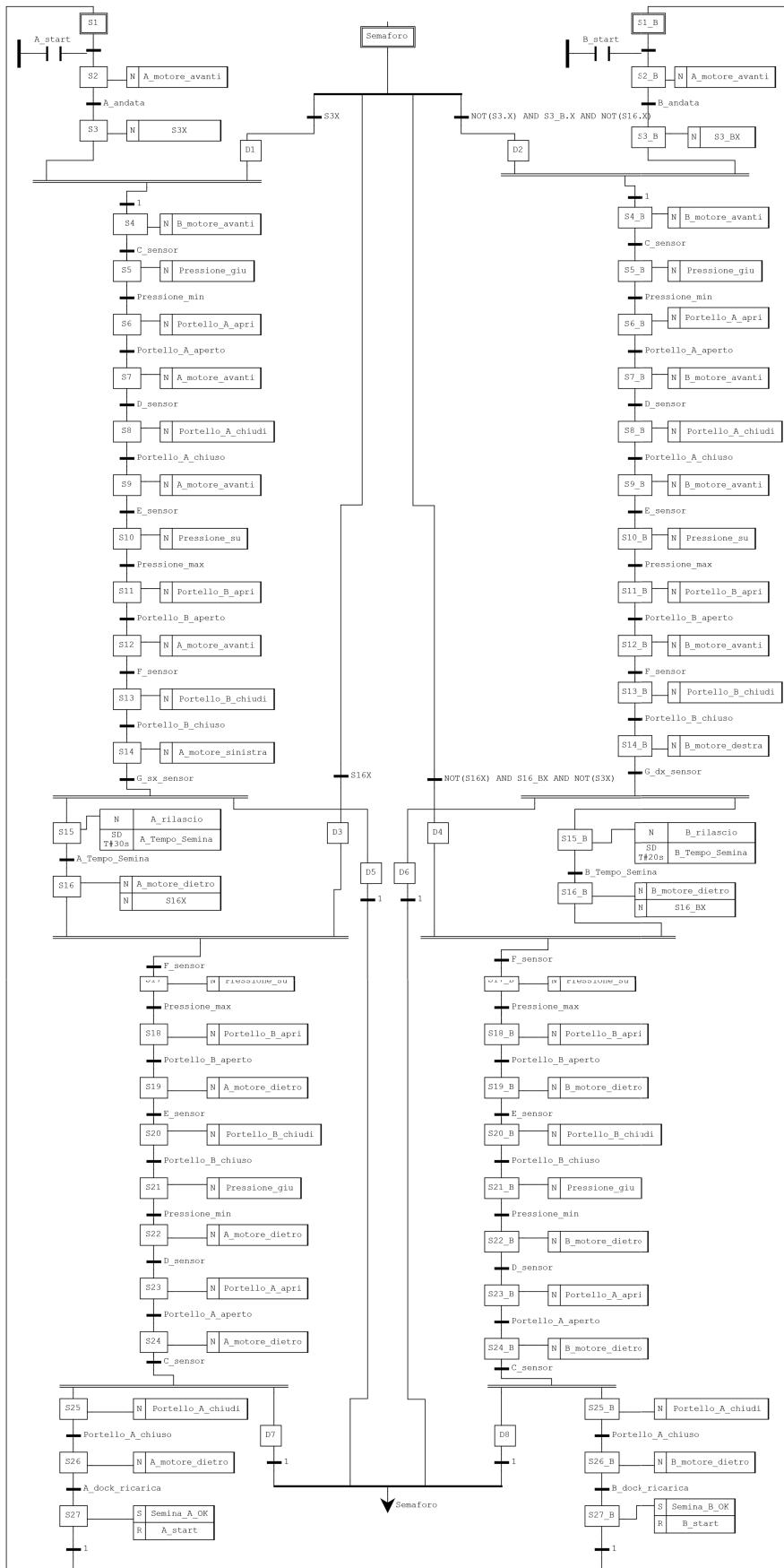


Figura 2.7: SFC della semina



Capitolo 3

Traduzione SFC in Ladder

A differenza del linguaggio SFC, il Ladder risulta essere uno strumento più universale data la sua compatibilità anche con PLC di vecchia generazione. Si suppone che per una missione su Marte si disponga di macchine di ultima generazione, tuttavia presentiamo una traduzione in Ladder come esercizio didattico.

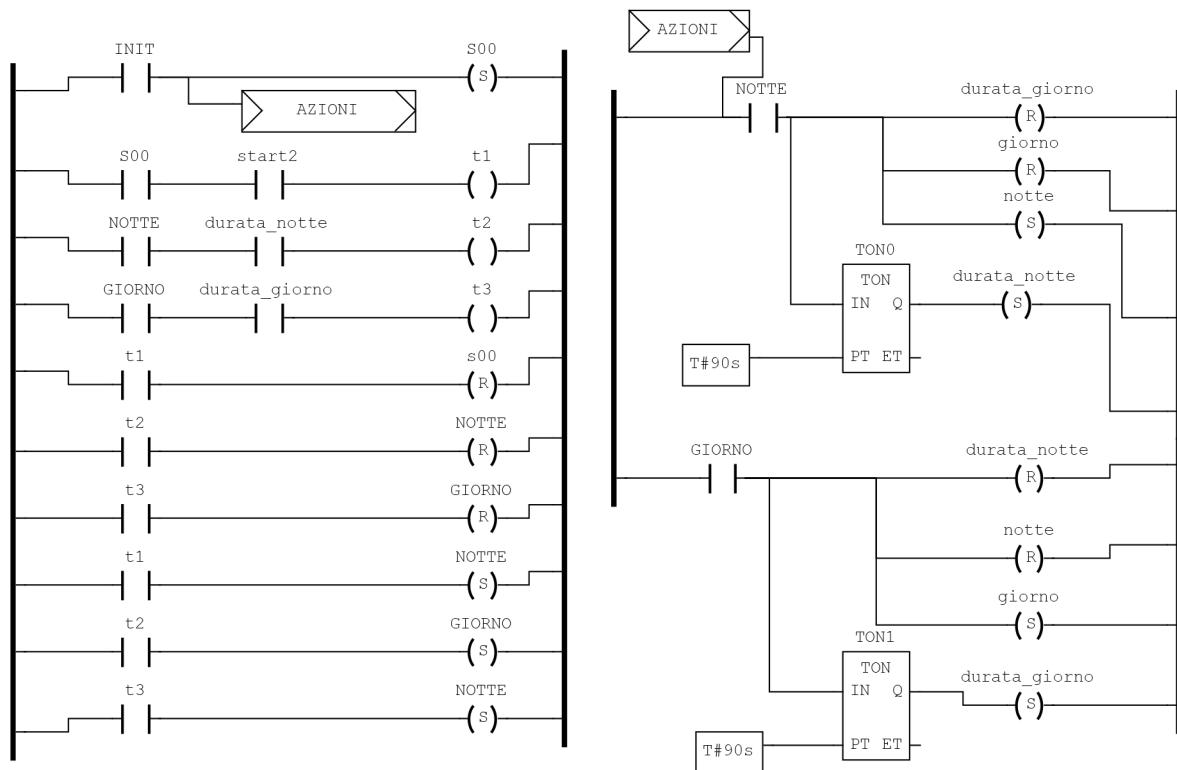


Figura 3.1: Ladder del ciclo Giorno Notte

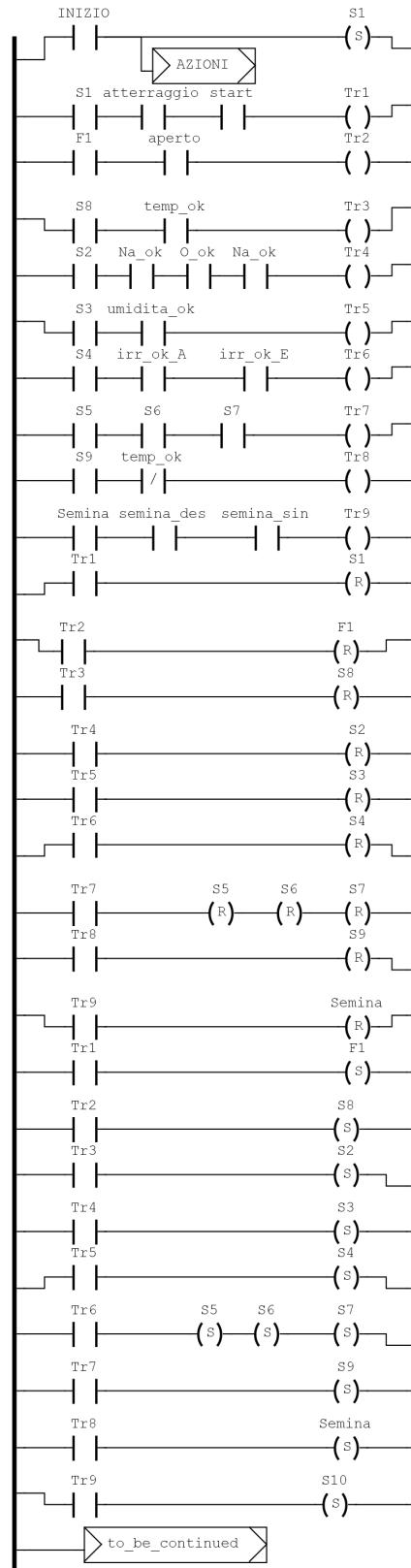


Figura 3.2: Ladder di inizializzazione - parte 1



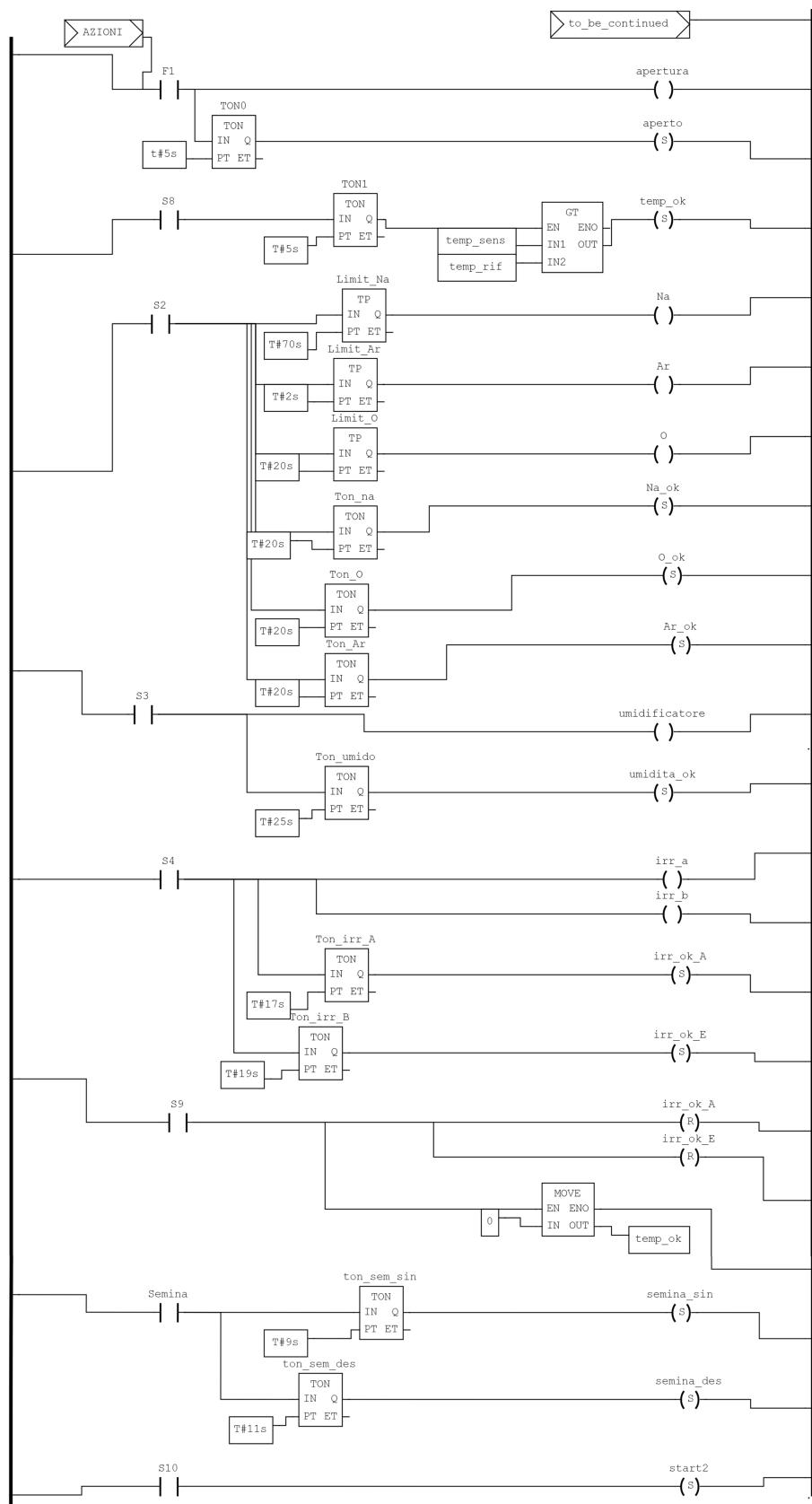


Figura 3.3: Ladder di inizializzazione - parte 2



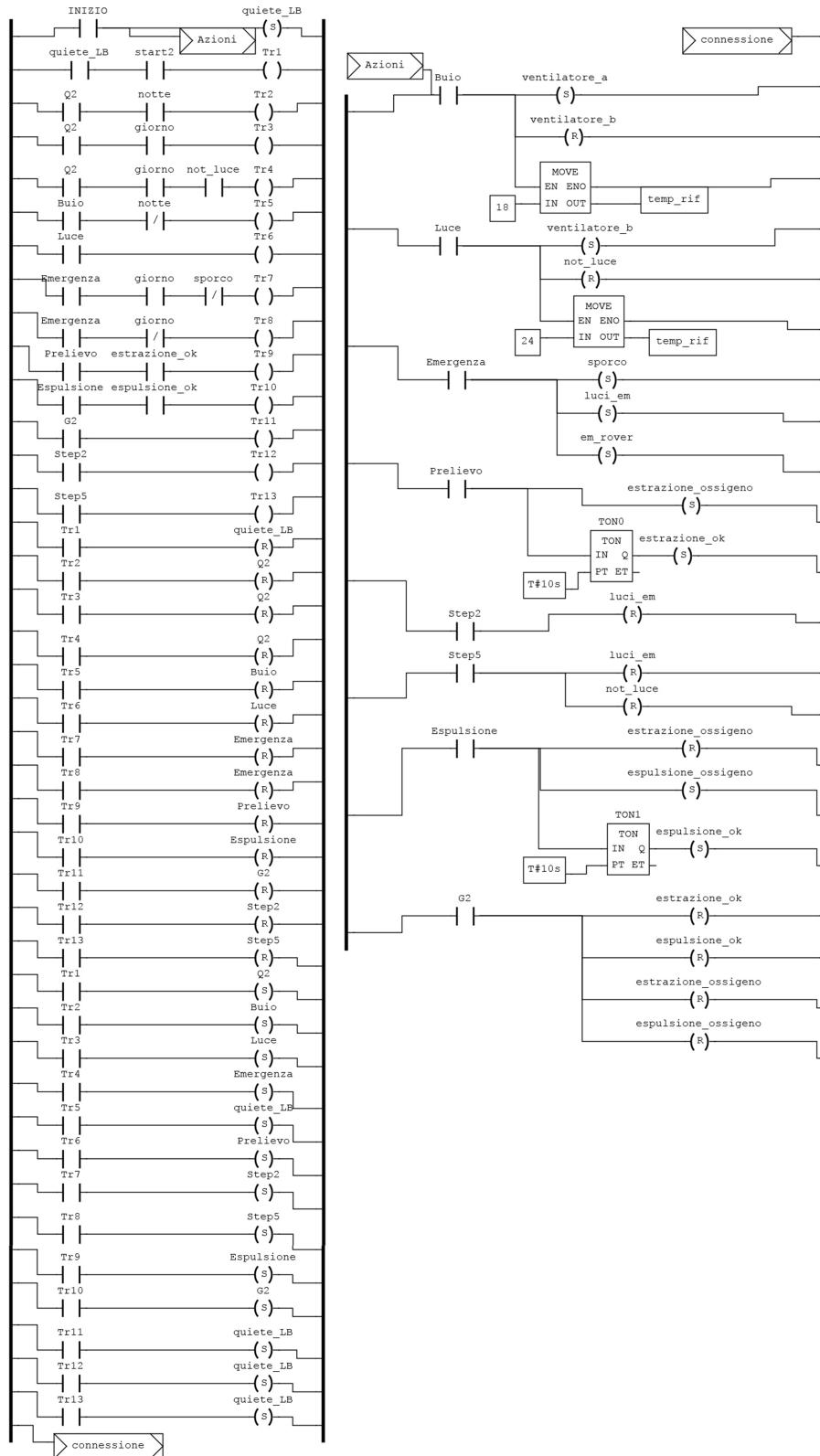


Figura 3.4: Ladder del ciclo luce buio



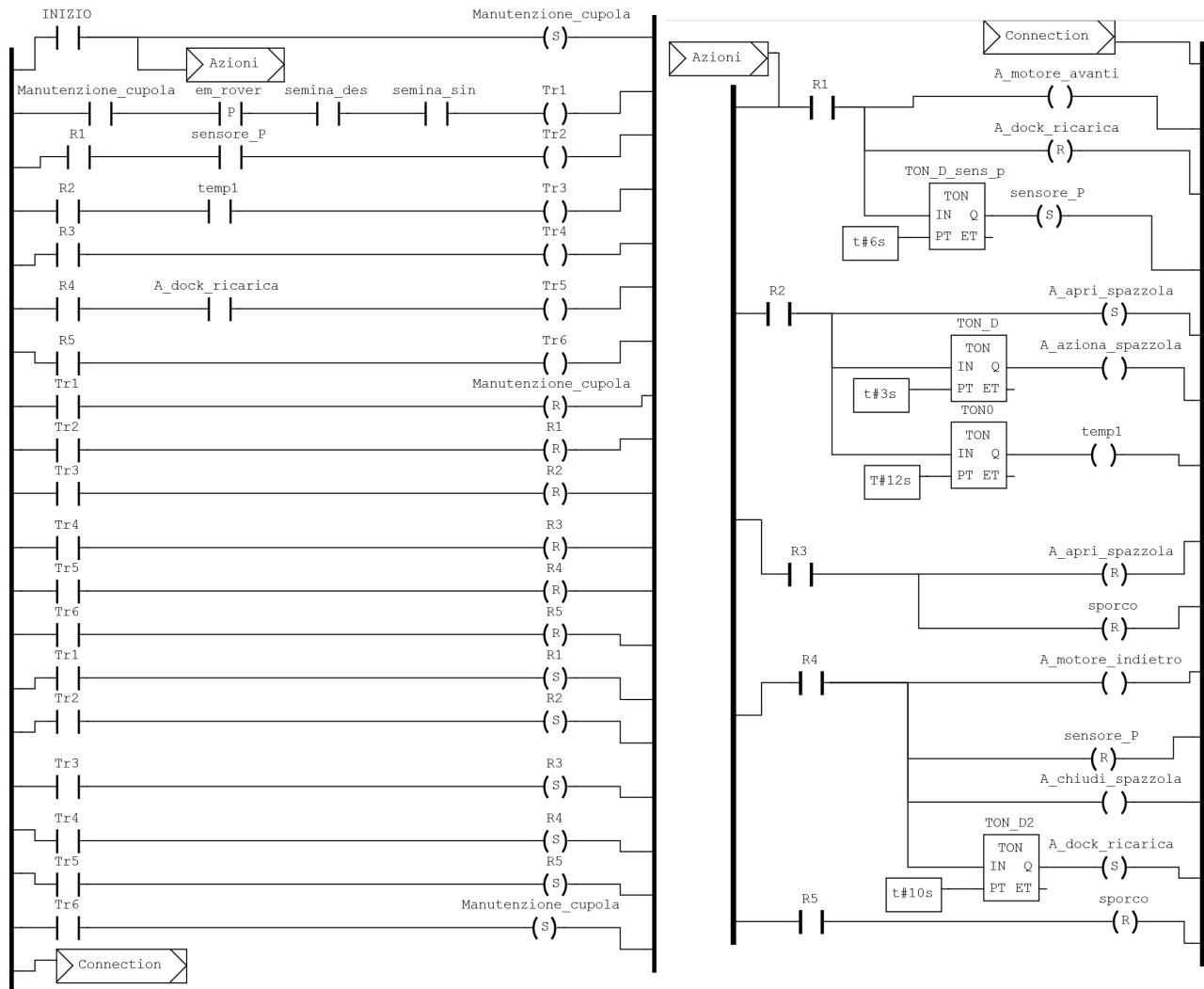


Figura 3.5: Ladder dell'emergenza Rover



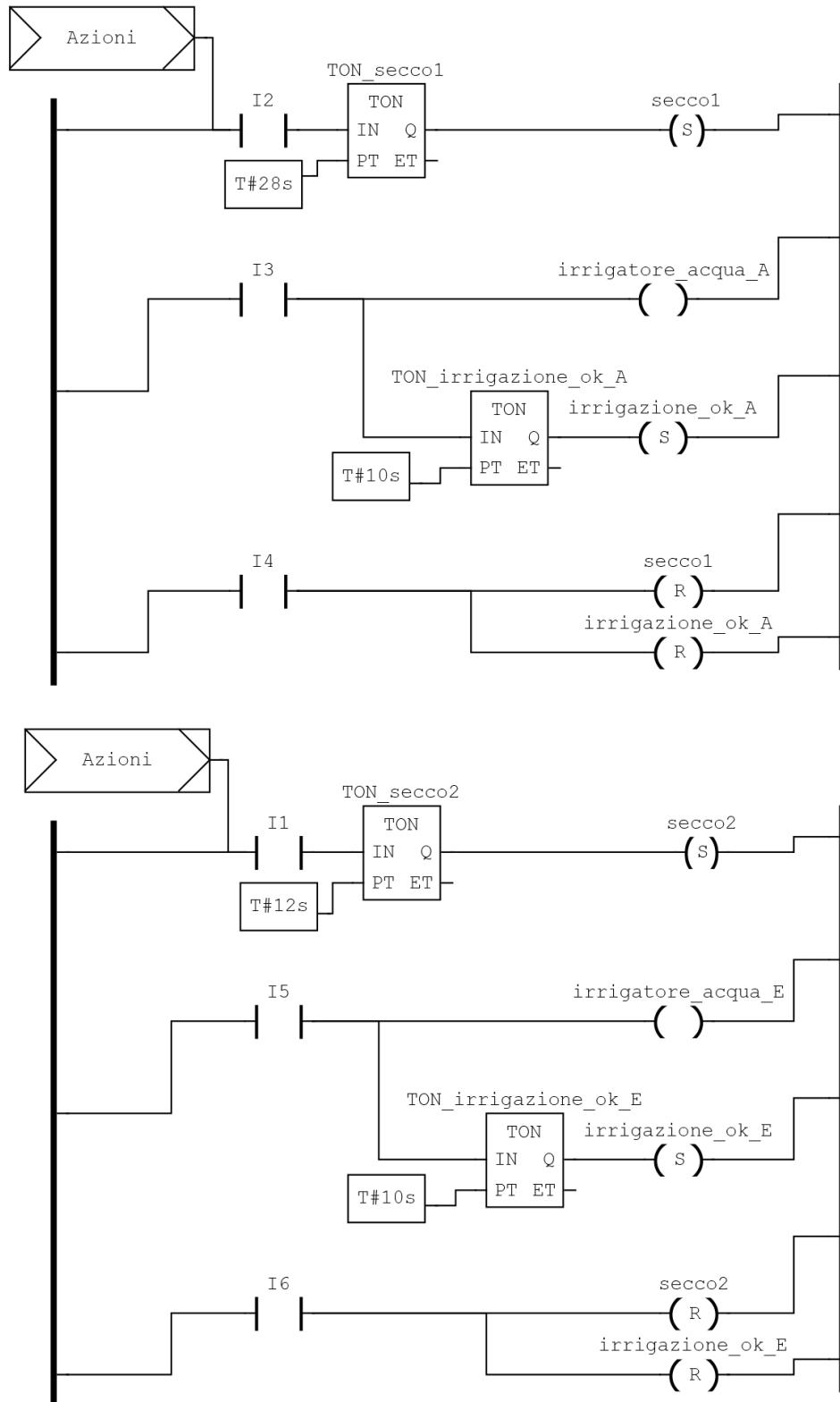


Figura 3.6: Ladder di irrigazione



Capitolo 4

Modello Termico

La temperatura del pianeta rosso varia dai -150C° nelle notti più fredde fino ai 20C° nei giorni più caldi. È necessario dunque un sistema di isolamento termico che garantisca una temperatura confortevole per la crescita dei vegetali all'interno della cupola. L'Aerogel ci è sembrato subito un ottimo candidato a questo scopo, con la sua conducibilità termica pari a 0.015 [W/mK] e la sua capacità di assorbire le radiazioni luminose, oltre ai suoi precedenti impieghi in ambito spaziale. Per irrobustire la capacità isolante di questo sistema si è scelto di impiegare anche il più comune gas Argon.

Per la modellazione di un sistema termico si consideri un ambiente delimitato da una superficie S composto da un materiale di conducibilità termica k . L'ambiente si troverà ad una temperatura T e sarà occupato da gas di capacità termica C . All'esterno dell'ambiente si registra una temperatura T_a . Voglio innalzare la temperatura interna T introducendo energia termica q .

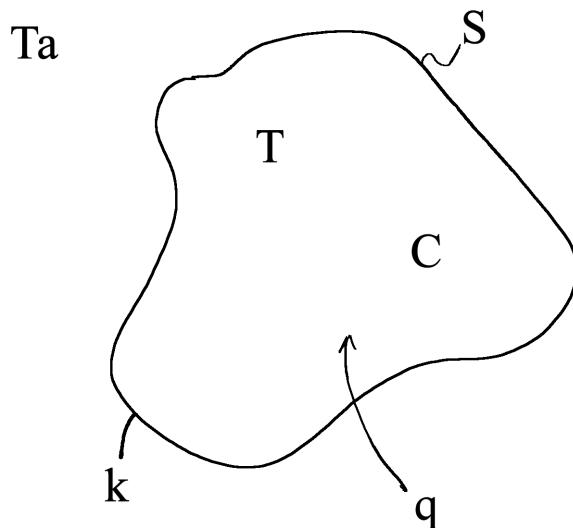


Figura 4.1: Modello Termico elementare

l'equazione di questo modello è

$$C \cdot \dot{T} = q - k(T - T_a)$$

Vogliamo progettare la nostra cupola servendoci di due strati di Aerogel ed un terzo strato di Lexan per consentire il passaggio di luce e la conservazione del calore, ma al contempo schermarci da radiazioni dannose. Le forti proprietà di isolamento termico dell'Aerogel saranno coadiuvate da quelle del gas Argon, di comune uso sulla terra per l'isolamento degli infissi. Gli strati di Aerogel e il Lexan saranno separati da uno spazio riempito di gas Argon, come illustrato dalla seguente figura:

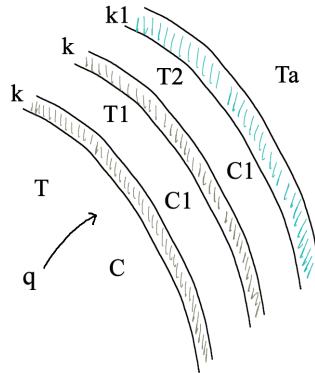


Figura 4.2: Modello termico della cupola

Le equazioni di questo modello termico saranno:

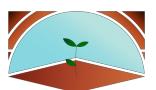
$$\begin{aligned} C \cdot \dot{T} &= q - k(T - T_1) \\ C_1 \cdot \dot{T}_1 &= -k_1(T_1 - T_2) + k(T - T_1) \\ C_1 \cdot \dot{T}_2 &= -k_2(T_2 - T_a) + k(T_1 - T_2) \end{aligned}$$

Controlleremo la temperatura interna alla cupola regolando l'afflusso di energia termica introdotta nel sistema per mezzo di una serie di resistenze riscaldanti, disseminate strategicamente lungo il perimetro interno della cupola. Pertanto ingressi, stati ed uscite saranno impostati nel seguente modo:

$$\begin{aligned} u_1 &= q, u_2 = T_a \\ x_1 &= T, x_2 = T_1, x_3 = T_2 \\ y &= T \end{aligned}$$

Fatte le precedenti osservazioni giungiamo al modello Ingresso-Stato-Uscita Lineare Tempo Invariante.

$$\begin{cases} \dot{x}_1 = \dot{T} = -\frac{k}{C}x_1 + \frac{k}{C}x_2 + \frac{1}{C}u_1 \\ \dot{x}_2 = \dot{T}_1 = -\frac{k^2}{C_1}x_1 + \frac{k^2-k}{C_1}x_2 + \frac{k}{C_1}x_3 \\ \dot{x}_3 = \dot{T}_2 = -\frac{kk_1}{C_2}x_2 + \frac{kk_1-k_1}{C_2}x_2 + \frac{k_1}{C_2}u_2 \\ y_1 = T = x_1 \end{cases}$$



In quanto modello lineare può essere ricavata una sua forma matriciale:

$$A = \begin{bmatrix} -\frac{k}{C} & \frac{k}{C} & 0 \\ -\frac{k^2}{C_1} & \frac{k^2-k}{C_1} & \frac{k}{C_1} \\ -\frac{kk_1-k_1}{C_2} & \frac{kk_1-k_1}{C_2} & 0 \end{bmatrix}, \quad B = \begin{bmatrix} -\frac{1}{C} & 0 \\ 0 & 0 \\ 0 & \frac{k_1}{C_2} \end{bmatrix}$$

$$C = [1 \ 0 \ 0] \quad D = [0 \ 0]$$

A questo punto non ci resta che riportare le Matrici del sistema in Matlab: generiamo un file cupola.m

Script Matlab

```

1 clear all
2 close all
3 clc
4
5 %suppongo massa 1kg per ogni gas
6 k=0.128; %coefficiente di scambio termico dell'Aerogel
7 k3=500; %coefficiente di scambio termico del Lexan
8 c=1030; %capacita termica di una generica atmosfera terrestre
9 Ca=0.52; %capacita termica del gas Argon
10
11 %Matrici del sistema
12 A=[-k/c k/c 0; -(k^2)/Ca k*(k-1)/Ca k/Ca; 0 -(k*k3)/Ca k3*(k-1)/Ca];
13 B=[1/c 0; 0 0; k3/Ca];
14 C=[1 0 0];
15 D=[0 0];
16
17 sys=ss(A,B,C,D);%interpreta le 4 matrici come un sistema dinamico
18
19 W=tf(sys); %ottengo le funzioni di trasferimento
20
21 t=(0:1e-2:2000)'; %scelgo un vettore dei tempi di 2000 secondi
22 r=24*ones(size(t)); %scelgo un riferimento costantemente pari a 24 C
23
24 %lancio la simulazione con simulink
25 out = sim('simulazione.slx');
26 uCLP = out.uCLP;
27 yCLP = out.yCLP;
28
29 figure(2)
30 plot(t,r,'k','linewidth',1)
31 plot(yCLP.time,yCLP.signals.values,'b','linewidth',2)
32 set(gcf,'Position',[100 100 750 350]);
33
34 figure(3), hold on, grid on
35 plot(t,r,'k','linewidth',1)
36 plot(uCLP.time,uCLP.signals.values,'r','linewidth',2)
37 plot(yCLP.time,yCLP.signals.values,'b','linewidth',2)
38 legend('riferimento', 'controllo PID', 'uscita')
39 title('Control action')
```



Lo script ci fornisce le due funzioni di trasferimento del sistema:

```
Command Window
>> W

W =

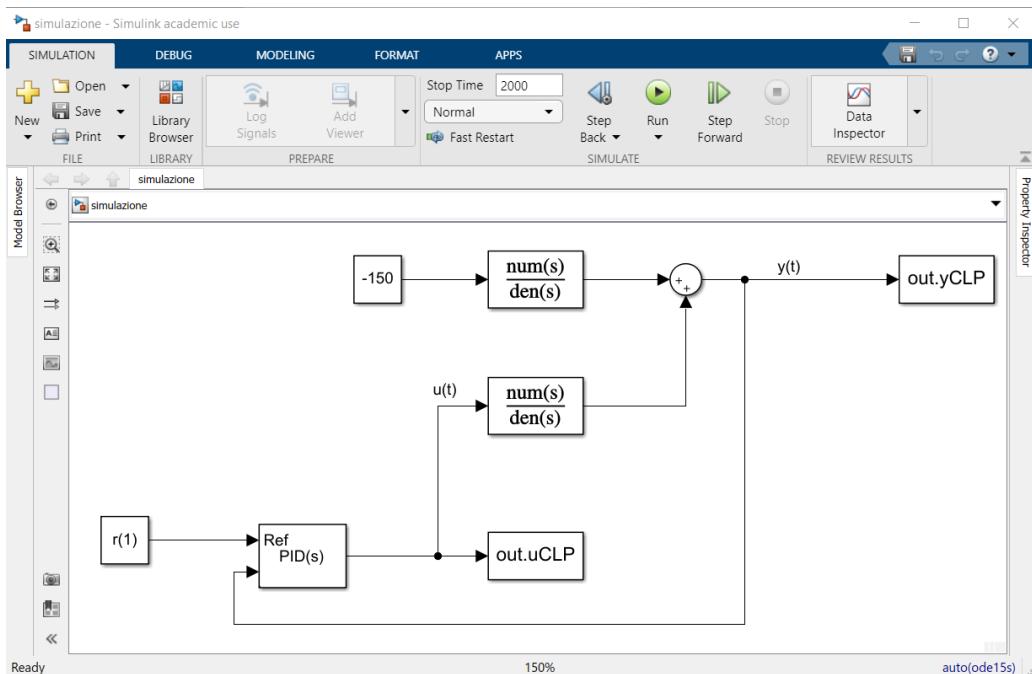
From input 1 to output:
0.0009709 s^2 + 0.8142 s + 0.2041
-----
s^3 + 838.7 s^2 + 210.4 s + 0.02941

From input 2 to output:
0.02941
-----
s^3 + 838.7 s^2 + 210.4 s + 0.02941

Continuous-time transfer function.
```

A questo punto ci spostiamo in Simulink generando un file "simulazione.slx". Il simulink mette a disposizione i blocchi Transfer Function in cui inseriamo i coefficienti delle funzioni di trasferimento ottenute.

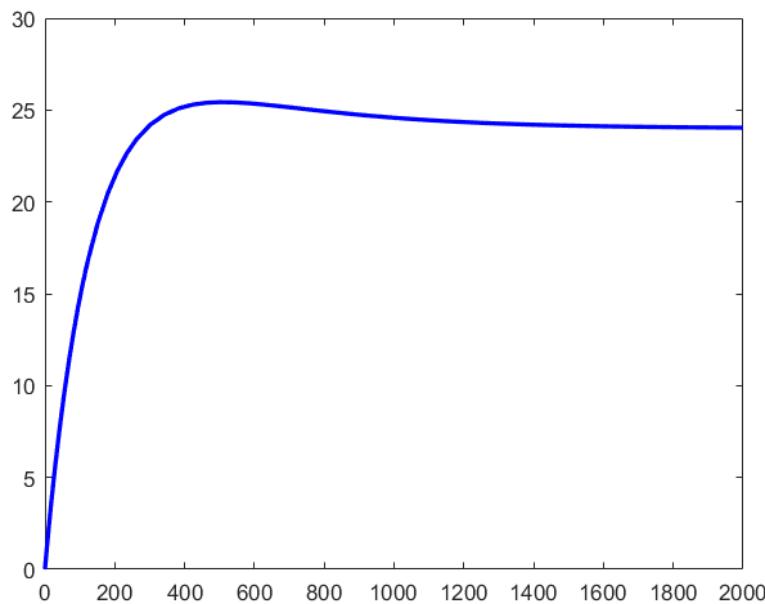
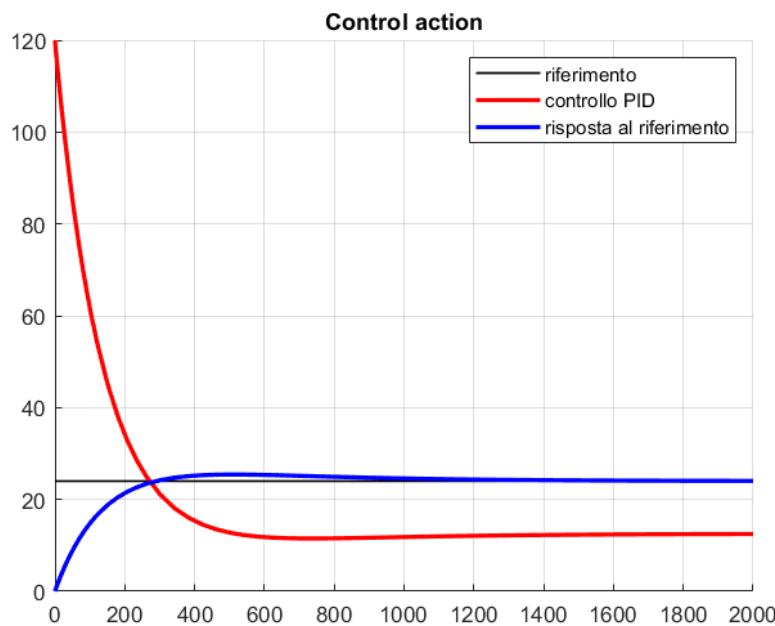
Consideriamo la funzione di trasferimento "from input 1 to output" come la parte del sistema che siamo in grado di controllare, poiché relativa all'ingresso $u_1 = q$. L'altra funzione di trasferimento rappresenta l'ingresso al sistema che non siamo in grado di controllare, ovvero la temperatura esterna alla cupola. Per questo motivo quest'ultima verrà considerata alla stregua di un disturbo, mentre la prima sarà collegata ad un regolatore PID standard fornito da Simulink.



Con un doppio clic su PID(s) ne apriamo le impostazioni e settiamo i parametri del controllore ai valori $P = 5$, $I = 0.01$, $D = 0$.

Diamo in ingresso al PID il riferimento scelto nel file "cupola.m" e colleghiamo in retroazione l'uscita dal nodo sommatore tra le due funzioni di riferimento, al PID.

A questo punto lanciamo l'esecuzione dello script che produrrà due grafici. L'asse orizzontale è l'asse dei tempi. In rosso troviamo l'andamento nel tempo dell'azione di controllo attuata dal PID, ovvero, l'andamento della potenza termica introdotta nella cupola. In blu vediamo l'andamento della temperatura nella cupola e in nero il valore di riferimento. Notiamo una leggera sovraelongazione che porta la temperatura ad un massimo di quasi 26°C . Per i nostri scopi non è un valore importante e pertanto possiamo lasciare il PID così impostato.



4.1 Comunicazione tra Simulink e OpenPLC Runtime

Una volta ottenuta la simulazione del nostro modello termico possiamo servircene per integrarla nel funzionamento del PLC che governa la cupola. Il modo previsto per effettuare questo collegamento non è tanto robusto e ci ha causato non pochi problemi: per questo vale la pena approfondire tutti i passaggi. Innanzitutto occorre allocare le variabili che conterranno il valore di riferimento "temp_rif" e la temperatura percepita nella cupola "temp_sens" come interi, rispettivamente in uscita (%QW0) e in ingresso (%IW0). Sul forum dedicato ad OpenPLC è presente una discussione a proposito della comunicazione tra Simulink e OpenPLC Runtime, reperibile a questo [link](#). Il protagonista di questa comunicazione è un programma C++ **SimLink** che stabilisce una connessione tra i due software per mezzo di un protocollo UDP.

Questo è il file di configurazione per l'interfaccia del programma SimLink utilizzato per la connessione tra OpenPLC e Simulink. Simlink deve conoscere l'indirizzo IP della macchina su cui girano i programmi e anche il tipo di variabili gestite da OpenPLC.

```
1 num_stations = "1" /*numero di stazioni previste*/
2 comm_delay = "100"
3
4 # -----
5 #   SIMULINK
6 # -----
7 simulink.ip = "localhost"
8
9 # -----
10 #   STATION 0
11 # -----
12 station0.ip = "localhost"
13 station0.add(analog_in) = "10001"
14 station0.add(analog_out) = "10002"
```

Per aggiungere variabili ad una stazione, basta scrivere il nome della stazione seguita dal comando "add" e dal tipo di variabile che si intende aggiungere. Il numero a destra dell'uguale si riferisce alla porta UDP utilizzata dalla connessione UDP lato Matlab/Simulink per mandare e ricevere le variabili. Per esempio:

```
station0.add(digital_out) = "10001"
```

Nel nostro caso abbiamo configurato il tipo di variabile come "analog_xx" poiché intendiamo trasmettere degli interi e non dei singoli bit. Le variabili aggiunte alla stazione saranno connesse al buffer di OpenPLC nell'ordine in cui appaiono: nel nostro caso analog_in alla porta 10001 sarà connesso alla variabile allocata in %IW0, analog_out alla porta 10002 sarà connesso alla variabile %QW0. Diversi tipi di variabili (digital_out, analog_in) vengono connessi a buffer differenti. Quindi se dopo un digital_out compare un analog_in questo sarà connesso a %IW0 (buffer di input analogico in posizione 0). Nel Main del programma simlink.cpp operiamo la seguente modifica così da vedere quali sono gli effettivi valori inviati e ricevuti.

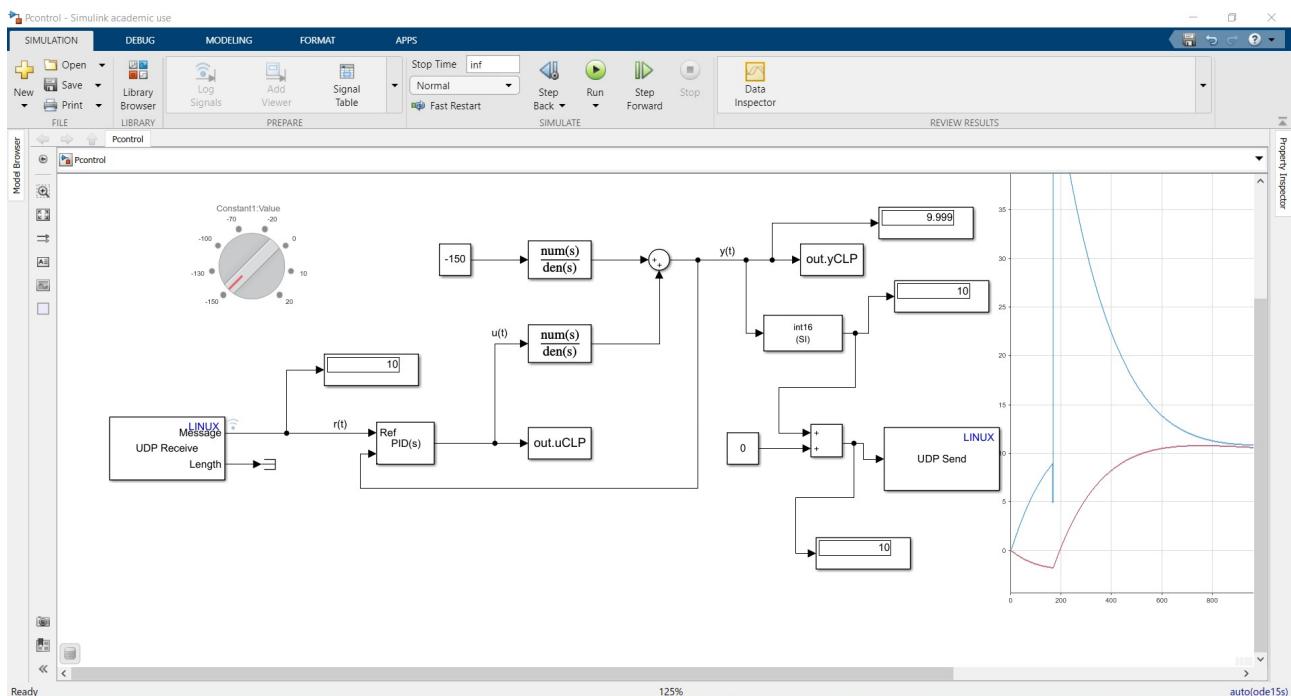
```
1 while(1)
2 {
3     pthread_mutex_lock(&bufferLock);
4     printf("\n-----\n");
5     printf("Station 1\nTemperatura di riferimento: %d\n", stations_data[0].analogOut[0]);
6     printf("Temperatura cupola: %d", stations_data[0].analogIn[0]);
7     pthread_mutex_unlock(&bufferLock);
8     sleep_ms(3000);
9 }
```



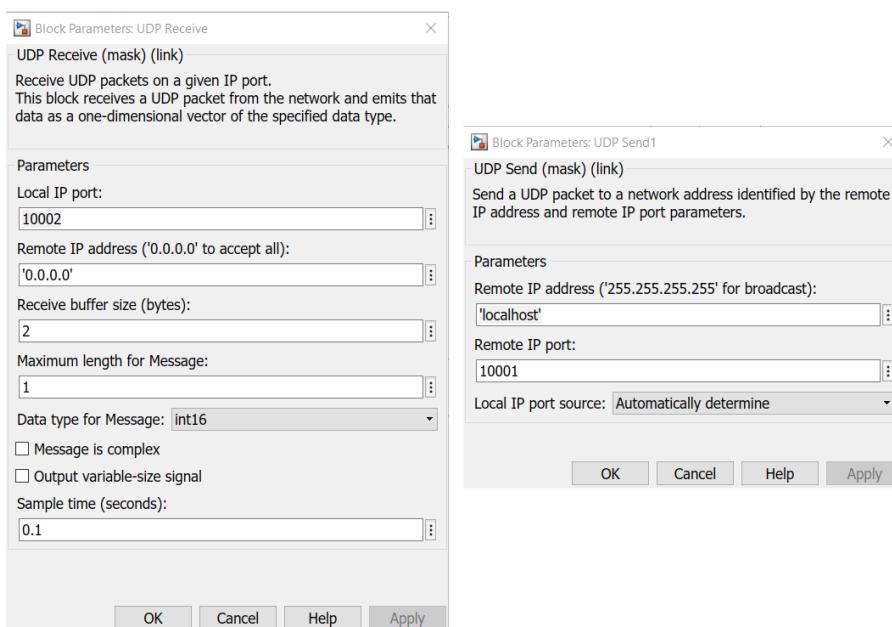
Adesso dobbiamo compilare il programma "ponte": se stiamo utilizzando Windows scarichiamo il software **CygWin**, avendo cura di selezionare il pacchetto lib gcc durante l'installazione. Eseguiamo CygWin e navighiamo fino alla cartella dove si troveranno i file interface.cfg e simlink.cpp. Compiliamo il programma con il comando:

```
g++ simlink.cpp -o simlink -pthread
```

Adesso dobbiamo completare la configurazione lato Simulink: aggiungiamo alla libreria l'add-on **Simulink Real Time**. Fatto questo possiamo copiare i blocchi UDP receive e UDP send dal programma Simulink di esempio disponibile [qui](#), e incollarli nel nostro "simulazione.slx", collegandoli in questo modo:



Ed avendo cura di settare i due blocchi UDP come illustrano i seguenti screenshot:





Nel programma Simulink abbiamo aggiunto una manopola per switchare tra diverse temperature possibili sul pianeta e testare il funzionamento in tutte le condizioni. Inoltre si è rivelato necessaria una conversione: l'uscita del nostro sistema viene generata come un numero reale, il PLC però gestisce degli interi, quindi abbiamo effettuato una conversione ad un intero a 16 bit. Nonostante ciò il valore veniva mal letto dal blocco UDP send e il PLC riceveva il valore sbagliato, pertanto abbiamo collegato l'uscita del blocco di conversione ad un blocco sommatore. Sommando uno zero costante abbiamo ottenuto una comunicazione pulita tra tutti i programmi.

Per la corretta esecuzione occore lanciare i programmi nell'ordine giusto: per prima cosa dobbiamo dare lo start ad OpenPLC Runtime, poi lanciamo la simulazione sul Simulink avendo cura di impostare sample time su inf, e infine lanciamo il programma ponte dal CygWin con il comando ”./simlink”. Dalla finestra CygWin è possibile leggere i valori scambiati. Nel nostro esempio non abbiamo ancora implementato un'interfaccia con ScadaBR e quindi il PLC resta fermo su una temperatura di riferimento di 10.

```
 /cygdrive/c/users/marcu/documents/SIMLINK

Station 1
Temperatura di riferimento: 10
Temperatura cupola: 6

Station 1
Temperatura di riferimento: 10
Temperatura cupola: 8

Station 1
Temperatura di riferimento: 10
Temperatura cupola: 10

Station 1
Temperatura di riferimento: 10
Temperatura cupola: 10

Station 1
Temperatura di riferimento: 10
Temperatura cupola: 11

Station 1
Temperatura di riferimento: 10
Temperatura cupola: 11
```

Notiamo che la temperatura nella cupola sale fino a 10, con una sovrelongazione che la porta a 11. Dopo qualche secondo il valore si assesterà a 10.



Capitolo 5

OpenPLC Runtime & ScadaBR

Per il funzionamento dell'SFC sono state allocate variabili per la maggior parte di tipo booleano, mentre la restante parte, relativa al controllo delle temperature, è di tipo intero. La variabile "temp_sens", in quanto variabile di ingresso intera, è stata allocata nella location %IW0. In questo registro, andremo a passare il valore della temperatura percepita all'interno della cupola. La variabile "temp_rif" sarà utilizzata per settare la temperatura di riferimento desiderata all'interno della cupola, e per questo sarà allocata come una variabile intera di uscita. Per il settaggio delle temperature di riferimento si è scelto di implementare delle azioni in linguaggio ST. L'azione "t_rif_XX" si occuperà di settare la temperatura di riferimento al valore desiderato tramite l'istruzione "temp_rif := XX;". L'altra azione utilizzata "t_okXX" si occupa di verificare che la temperatura percepita all'interno della cupola abbia raggiunto il valore di riferimento tramite l'istruzione "temp_ok := GE (temp_sens, XX);". Al termine della progettazione dell'SFC, il file compilato "marte.st" andrà caricato ed eseguito su OpenPLC Runtime. Tramite la funzione di "monitoring" sarà possibile verificare il corretto funzionamento del programma. Per interfacciarsi col programma, come anticipato precedentemente, ci serviremo del software Open Source ScadaBR. Una volta lanciato ScadaBR su una virtual machine ed effettuato l'accesso, ci occuperemo di far comunicare l'interfaccia col programma in esecuzione su OpenPLC Runtime. Da ScadaBR instanzieremo una nuova data source che si servirà del protocollo Modbus IP. Selezioneremo un Transport Type di tipo "TPC with keep alive" scegliendo come Host l'indirizzo IP della macchina su cui gira OpenPLC Runtime. Collegheremo le variabili di OpenPLC Runtime come mostrato nella figura nella pagina seguente.

#	Nome	Classe	Yipo	Location	Initial Value	Opzione	Documentazione
1	temp_sens	Locale	INT	%IW0	0		temperatura percepita dal sensore
2	temp_rif	Locale	INT	%QW0	0		temperatura di riferimento
3	atterraggio	Locale	BOOL	%QX0.0	0		avvenuto atterraggio
4	start	Locale	BOOL	%QX0.1	0		start dalla terra
5	temp_ok	Locale	BOOL	%QX3.0	0		temperatura al livello desiderato
6	giorni	Locale	INT		0		
7	Na	Locale	BOOL		0		eroga azoto
8	Ar	Locale	BOOL		0		eroga argon
9	O	Locale	BOOL		0		eroga ossigeno
10	Na_ok	Locale	BOOL	%QX0.2	0		azoto al livello desiderato
11	Ar_ok	Locale	BOOL	%QX0.3	0		argon al livello desiderato
12	O_ok	Locale	BOOL	%QX0.4	0		ossigeno al livello desiderato
13	pompa_acqua_A	Locale	BOOL		0		aziona pompa a
14	pompa_acqua_E	Locale	BOOL		0		aziona pompa b
15	umidita_ok	Locale	BOOL	%QX0.5	0		umidità al livello desiderato
16	irrigazione_ok_A	Locale	BOOL	%QX0.6	0		irrigazione preliminare alora avvenuta
17	irrigazione_ok_E	Locale	BOOL	%QX0.7	0		irrigazione preliminare edera avvenuta
18	semina_des	Locale	BOOL	%QX1.0	0		semina edera completata
19	semina_sin	Locale	BOOL	%QX1.1	0		semina aloe completata
20	sens_luce_buio	Locale	BOOL	%QX1.2	0		variabile simulazione giorno notte
21	irrigatore_acqua_A	Locale	BOOL		0		irrigatore a attivo
22	irrigatore_acqua_E	Locale	BOOL		0		irrigatore e attivo
23	start2	Locale	BOOL	%QX3.1	0		secondo segnale di start
24	T_18	Locale	INT		17		
25	T_24	Locale	INT		23		
26	not_luce	Locale	BOOL	%QX1.3	0		luce non percepita nella cupola
27	luci_em	Locale	BOOL	%QX1.4	0		illuminazione ausiliaria
28	em_rover	Locale	BOOL	%QX1.5	0		emergenza rover
29	fine_em_luce	Locale	BOOL	%QX1.6	0		
30	estrazione_ok	Locale	BOOL	%QX1.7	0		avvenuta estrazione di ossigeno
31	A_Motore_avanti	Locale	BOOL	%QX3.6	0		
32	A_Motore_indietro	Locale	BOOL		0		
33	A_apri_spazzola	Locale	BOOL	%QX4.0	0		apertura spazzola rover
34	A_chiudi_spazzola	Locale	BOOL		0		
35	A_dock_ricarica	Locale	BOOL	%QX2.0	1		rover in posizione di ricarica presso il dock
36	secco1	Locale	BOOL	%QX2.1	0		terreno aloe secco
37	secco2	Locale	BOOL	%QX2.2	0		terreno edera secco
38	A_aziona_spazzola	Locale	BOOL	%QX2.7	0		aziona spazzola (dopo apertura)
39	sensore_P	Locale	BOOL	%QX3.7	0		sensore perimetrale
40	espulsione_ok	Locale	BOOL	%QX2.3	0		avvenuta espulsione ossigeno
41	pulito	Locale	BOOL	%QX2.4	1		stato di pulizia della cupola
42	estrazione_ossigeno	Locale	BOOL	%QX2.5	0		estrazione in corso
43	espulsione_ossigeno	Locale	BOOL	%QX2.6	0		espulsione in corso
44	ventilatore_A	Locale	BOOL	%QX3.4	0		attiva ventilatore a
45	ventilatore_B	Locale	BOOL	%QX3.5	0		attiva ventilatore b

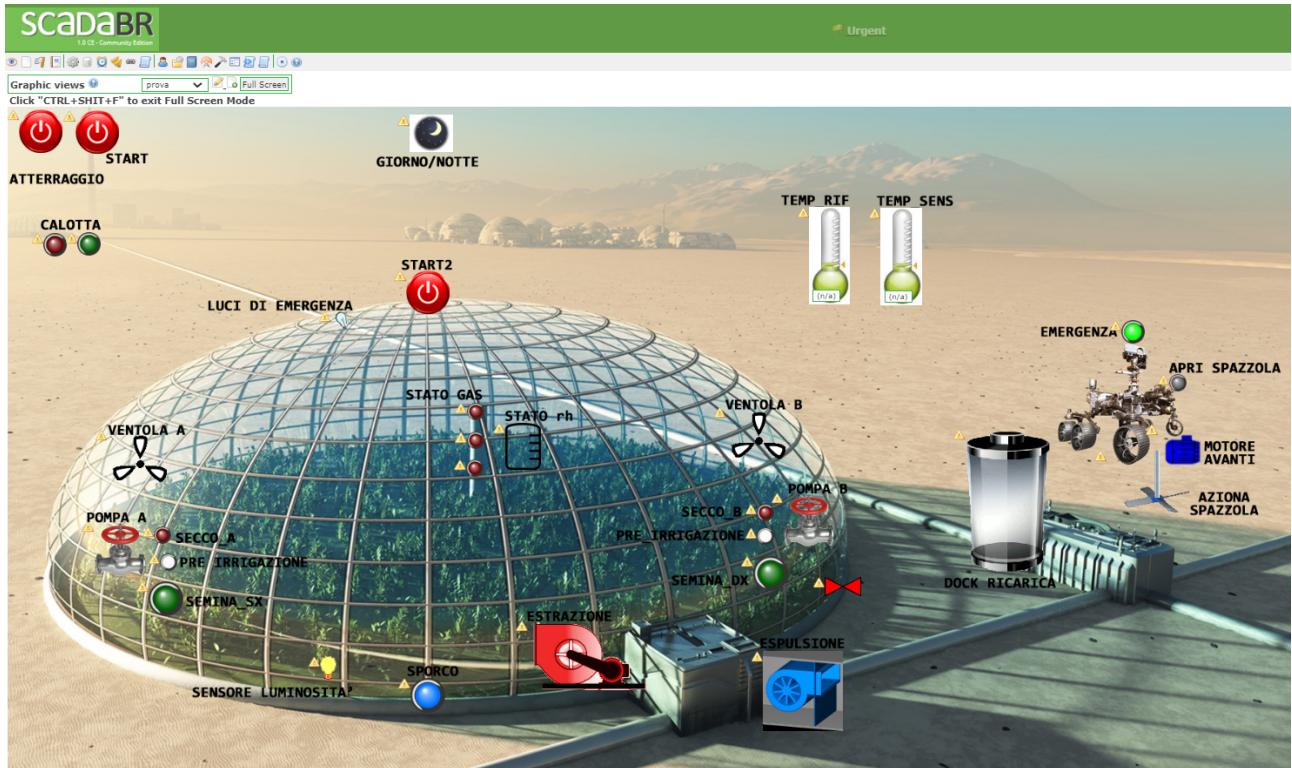


ScadaBR fa riferimento alle variabili del programma allocate in ordine crescente, ad esempio alla variabile allocata in %QX0.0 corrisponderà l'offset 0, alla variabile allocata in %QX0.1 corrisponderà l'offset 1 e così via. Vale la pena specificare che, siccome OpenPLC gestisce registri fino a %QXY.7, i registri 8 e 9 non saranno utilizzati, tuttavia, questo limite non riguarderà ScadaBR che proseguirà con l'assegnazione degli offset in modo continuo. La corrispondenza tra le variabili del programma e degli offset in ScadaBR avverrà come illustrato nella seguente immagine:

Points						
Name	Data type	Status	Slave	Range	Offset (0-based)	
A_apri_spazzola	Binary		1	Coil status	32	
A_aziona_spazzola	Binary		1	Coil status	23	
A_dock_ricarica	Binary		1	Coil status	16	
A_motore_avanti	Binary		1	Coil status	30	
aperto	Binary		1	Coil status	38	
apertura	Binary		1	Coil status	37	
Ar	Binary		1	Coil status	3	
atterraggio	Binary		1	Coil status	0	
em_rover	Binary		1	Coil status	13	
espulsione_ok	Binary		1	Coil status	19	
espulsione_oss	Binary		1	Coil status	22	
estrazione_ok	Binary		1	Coil status	15	
estrazione_oss	Binary		1	Coil status	21	
fine_em_luce	Binary		1	Coil status	14	
giorno	Binary		1	Coil status	33	
irrigatore_A	Binary		1	Coil status	35	
irrigatore_B	Binary		1	Coil status	36	
irrigazione_ok_A	Binary		1	Coil status	6	
irrigazione_ok_B	Binary		1	Coil status	7	
luci_em	Binary		1	Coil status	12	
Na	Binary		1	Coil status	2	
not_luce	Binary		1	Coil status	11	
notte	Binary		1	Coil status	34	
O	Binary		1	Coil status	4	
secco1	Binary		1	Coil status	17	
secco2	Binary		1	Coil status	18	
semina_des	Binary		1	Coil status	8	
semina_sin	Binary		1	Coil status	9	
sens_giorno	Binary		1	Coil status	10	
Sensore_P	Binary		1	Coil status	31	
sporco	Binary		1	Coil status	20	
start	Binary		1	Coil status	1	
start2	Binary		1	Coil status	25	
temp_ok	Binary		1	Coil status	24	
temp_rif	Numeric		1	Holding register	0	
temp_sens	Numeric		1	Input register	0	
umidita_ok	Binary		1	Coil status	5	
ventilatore_A	Binary		1	Coil status	28	
ventilatore_B	Binary		1	Coil status	29	



Tutte le variabili avranno un range coil status, poiché variabili di uscita di tipo booleano, mentre "temp_rif" e "temp_sens" avranno rispettivamente range holding register e input register. ScadaBR mette a disposizione una vasta scelta di spie luminose per la segnalazione dello stato delle variabili. Una volta scelta un'immagine di background abbiamo disposto tali spie come illustrato in figura:



5.1 Conclusioni

Lo scopo del progetto e le soluzioni adottate rispondono a dei canoni molto ambiziosi, e pertanto la trattazione di stampo didattico intrapresa in questo lavoro, trascura molti aspetti di fondamentale importanza alla realizzazione effettiva di un modulo di terraformazione. In quanto studenti, gli strumenti utilizzati ci hanno avvicinato ad un ambiente lavorativo affine all'automazione industriale, affrontando problematiche fantasiose ed utilizzando soluzioni reali.

