Università degli Studi di Perugia

Dipartimento di Matematica e Informatica

Report di Laboratorio
Sistemi Elettronici e Sensori per l'Informatica

Studenti

Dedi Vittorio
Palladino Danilo

Anno Accademico 2024-2025

Sommario

Introduzione

Per questa esercitazione analizzeremo i dati di un test TVAC (Test in camera a vuoto termico). Questi sono stati registrati in un file di log contenente molteplici parametri, tra cui: temperature, pressioni, impostazioni di regolazione e stati dei sistemi. L'obiettivo è analizzare e visualizzare questi dati per identificare dei comportamenti specifici tramite dei plot, facendo così una piccola analisi dei dati. L'esercitazione è interamente scritta in linguaggio Python, questo ci permetterà di importare, organizzare e analizzare i dati contenuti nel file di test del TVAC. Andremo a generare dei grafici rappresentativi tramite l'utilizzo di librerie dedicate.

La nostra scelta è stata quella di dividere in 3 capitoli e di seguito andremo a spiegare brevemente di cosa tratta ognuno di essi.

Il primo capitolo introduce il contesto del test TVAC, spiegandone il funzionamento e l'importanza di verificare le prestazioni dei componenti spaziali. Si farà poi una panoramica delle condizioni simulate, come il vuoto ($5x10^{-5}$ mbar) e le grandi variazioni di temperatura (da -100 °C a 125 °C). Inoltre, verrà brevemente descritto il contenuto del file di log analizzato nell'esercitazione.

Nel secondo capitolo viene presentato il codice sviluppato, con un focus sulla spiegazione delle funzioni e sulla definizione delle classi. Il capitolo illustra la struttura del programma e le sue funzionalità.

Il terzo ed ultimo capitolo analizza i grafici richiesti, In particolare: Il grafico della temperatura dello shroud e del suo setpoint rispetto al tempo, il grafico delle pressioni della camera in funzione del tempo e infine il grafico che evidenzia i momenti in cui la temperatura dello shroud aumenta, diminuisce o resta stabile.

Capitolo 1: Introduzione TVAC

La camera a vuoto termico (TVAC) è uno strumento tecnologico avanzato, indispensabile per testare componenti e sistemi destinati all'ambiente spaziale. Essa è progettata per simulare le condizioni estreme che i dispositivi incontreranno nello spazio, offrendo un ambiente controllato che riproduce sia il vuoto che le estreme escursioni termiche tipiche delle missioni spaziali. Le rilevazioni analizzate in questa esercitazione sono state effettuate utilizzando il TVAC dell'Università di Perugia, situato presso la sede di Terni. Questo offre ottime capacità di simulazione e raccolta dei dati, risultando ideale per attività di ricerca e sviluppo.

La Figura 1-1 illustra l'aspetto esterno della camera TVAC, mostrando la struttura esterna e i sistemi utilizzati per generare le condizioni necessarie ai test.

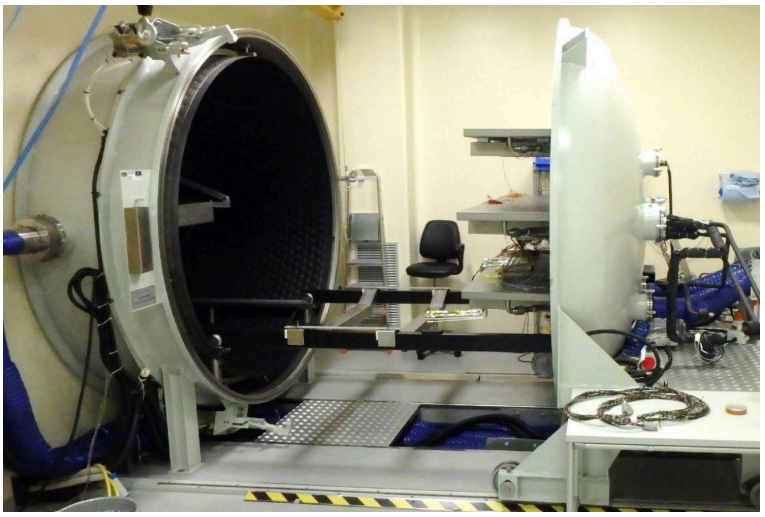

Figura 1-1

Il test TVAC viene condotto in condizioni di vuoto, con una pressione interna che raggiunge i $5x10^{-5}$ mbar, un valore che simula efficacemente l'assenza di atmosfera nello spazio. Allo stesso tempo, le temperature all'interno della camera oscillano tra -100 °C e 125 °C, riproducendo le escursioni termiche estreme a cui i componenti spaziali possono essere sottoposti. In questo caso abbiamo la possibilità di controllare separatamente la temperatura dello shroud (il rivestimento interno della camera) e delle piastre radianti (cold plates).

Come mostrato nella figura 1-2, queste piastre, disposte in tre strati distinti e consentono di simulare con elevata precisione le condizioni operative dei componenti, regolando localmente il trasferimento termico.

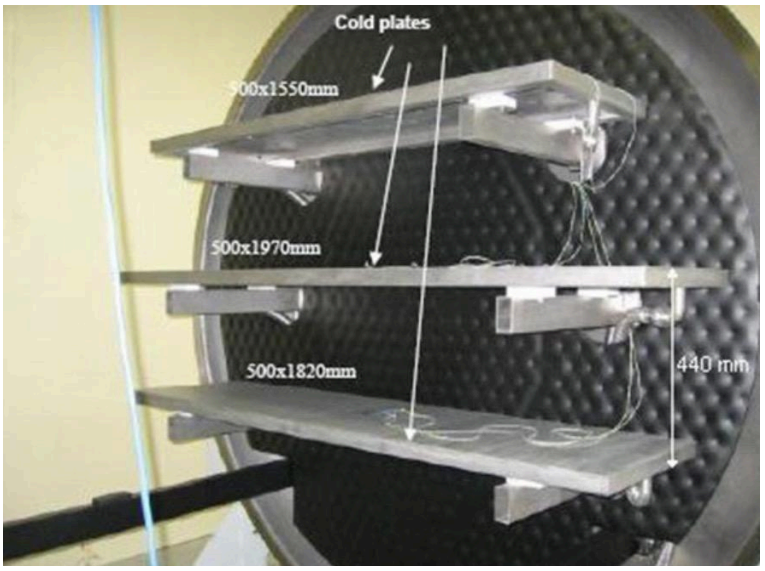

Figura 1-2

Durante i test, un’enorme quantità di dati viene raccolta localmente e salvata in un file di log strutturato in 70 colonne. Ogni colonna registra una specifica variabile monitorata, alcune tra queste sono: Data e ora, Temperatura dello shroud, Temperatura del fluido nelle cold plates, Pressione interna alla camera. Tutti questi valori ci servono per garantire che le condizioni di vuoto siano mantenute stabili e coerenti con i requisiti del test, tramite questi possiamo anche capire in caso di malfunzionamenti dove è necessario agire per riportare tutto ad uno stato di equilibrio.

Capitolo 2: Implementazione del codice

Come accennato in precedenza, l'obiettivo principale di questa esercitazione è analizzare i dati contenuti nel file di log generato durante il test TVAC. Per raggiungere questo scopo, è stato necessario sviluppare un programma in Python che ci ha permesso di leggere, elaborare e manipolare i dati in modo efficiente. Per la buona riuscita del progetto, non utilizzeremo solo Python, ma anche alcune librerie che faciliteranno operazioni fondamentali come l'analisi dei dati, la gestione dei file e la visualizzazione grafica.
In particolare, creeremo una classe Python denominata **`TvacTest`** mentre all'interno della classe saranno definiti diversi metodi, ciascuno con funzioni specifiche, ad esempio: Un metodo per plottare i grafici dei dati rilevati e un metodo per determinare i trend nelle variazioni dei dati, consentendo di identificare se i valori mostrano un aumento, una diminuzione o una stabilità.
I grafici richiesti dovranno rappresentare in modo chiaro i seguenti aspetti: Temperatura dello shroud (T Shroud) e il relativo setpoint rispetto al tempo, Pressioni della camera (P Full Range 1 chamber-ITR90 e P Chamber) in funzione del tempo e gli Intervalli di variazione della T Shroud, evidenziando quando questa è crescente, decrescente o stabile.

```
import numpy as np
import matplotlib.pyplot as plt
from datetime import datetime

header_to_skip = 155
file_path = "ESERCITAZIONI/ESERCITAZIONE TVAC/TVAC.txt"

class TvacTest():
   def __init__(self):
       self.columns = {}
   def create_columns(self):
       column_names = (np.genfromtxt(file_path, skip_header =
header_to_skip, max_rows = 1, delimiter="\t", dtype=str)).tolist()

       for i in range(len(column_names)):
           self.columns[column_names[i]] = (np.genfromtxt(file_path,
skip_header = header_to_skip+1, usecols=(i), delimiter="\t",
dtype=str)).tolist()
       return self.columns

   def trend(self, data, column_name):
       values = data[column_name]
       response = []
```

```
        if column_name == "Date Time":
            values = [datetime.strptime(data_str, "%d/%m/%y %H:%M:%S")
for data_str in values]
            return values
        else:
            values = np.asarray(data[column_name]).astype(float)

            for element in range(len(values) - 1):
                if (-0.099 <= values[element] - values[element+1] <=
0.099):
                    response.append(0) #stabile
                elif (values[element] - values[element+1] > 0.099):
                    response.append(-20) #decresce
                elif (values[element] - values[element+1] < -0.099):
                    response.append(20) #aumenta
            return response

    def create_array(self, data, column_name):
        values = data[column_name]

        if column_name == "Date Time":
            values = [datetime.strptime(data_str, "%d/%m/%y %H:%M:%S")
for data_str in values]
            return values
        else:
            values = np.asarray(data[column_name]).astype(float)
            return values

def plot_data(x_data, y_data, x_label="X", y_label="Y",
title="Grafico"):
    plt.plot(x_data, y_data, linestyle='-', color='b')
    plt.xlabel(x_label)
    plt.ylabel(y_label)
    plt.title(title)
    plt.grid(True)
    plt.show()

def comparison(value1, value2, label1, label2, scale = False):
    plt.figure(figsize=(12, 6))
    plt.plot(date_time, value1, linestyle='-', color='b', label=label1)
    plt.plot(date_time, value2, linestyle='-', color='r', label = label2)
    plt.xlabel('Date Time')
    plt.ylabel(f"{label1} vs {label2}")
    if scale:
        plt.yscale("log")
```

```
    plt.legend()
    plt.grid(True)
    plt.show()

tvac = TvacTest()
data = tvac.create_columns()

trend = tvac.trend(data, "T Shroud")
date_time = tvac.trend(data, "Date Time")
plot_data(date_time[:-1], trend, x_label="Date Time", y_label= "T
Shroud", title="Trend T Shroud")

t_shroud = tvac.create_array(data, "T Shroud")
t_shroud_setpoint = tvac.create_array(data, "T Shroud Setpoint")
comparison(t_shroud, t_shroud_setpoint, "t_shroud", "t_shroud_setpoint")

P_Full_Range_1_chamber_ITR90 = tvac.create_array(data, "P Full Range 1
chamber-ITR90")
P_Chamber = tvac.create_array(data, "P Chamber")
comparison(P_Full_Range_1_chamber_ITR90, P_Chamber, "P Full Range 1
chamber-ITR90", "P_Chamber", scale=True)
```

Inizialmente, definiamo alcune variabili globali fondamentali necessarie per il funzionamento. La variabile **`header_to_skip`** indica quante righe devono essere ignorate all'inizio del file di log per escludere le intestazioni e altre informazioni preliminari non necessarie all'analisi. La variabile **`file_path`** invece, specifica il percorso del file contenente i dati da elaborare, che rappresenta la base su cui si fondano tutte le operazioni successive.
La classe **`TvacTest`** rappresenta il nucleo operativo del programma, questa contiene una serie di metodi che saranno necessari per effettuare l'analisi dei dati.

Al momento della creazione di un'istanza della classe, il costruttore inizializza un dizionario vuoto chiamato **`self.columns`**, dove verranno memorizzati i dati letti dal file di log. Il metodo **`create_columns`** legge il file di log utilizzando **`NumPy`** per estrarre inizialmente nomi delle colonne e successivamente i relativi valori. Ogni colonna viene aggiunta al dizionario, con il nome della colonna come chiave e i dati associati come valore. Il dizionario risulta così essere una struttura facilmente accessibile, che ci consentirà di riferirci a qualsiasi colonna del file in modo diretto.

Il metodo trend ha come funzione principale quella di analizzare l'andamento dei valori di una specifica colonna per determinare se ci sono variazioni significative tra i dati consecutivi. Se la colonna in esame contiene dati temporali, come una colonna di data e ora, questi

vengono convertiti in oggetti timestamp per consentire calcoli precisi, mentre per tutte le restanti colonne, il metodo calcola la differenza tra ogni valore e il successivo, classificando queste variazioni in tre categorie: stabilità, aumento o diminuzione.

Ogni categoria è rappresentata da un valore numerico specifico, nel nostro caso (0, -20 e 20), ciò facilita l'elaborazione e l'interpretazione dei risultati. Oltre a questo, nel codice è incluso un metodo chiamato **`create_array`**, che consente di estrarre i valori di una colonna e convertirli in array **`NumPy`** o in liste di oggetti datetime, a seconda della natura dei dati.
Per quanto riguarda la rappresentazione visiva, nel codice abbiamo incluso due funzioni esterne alla classe: **`plot_data`** e **`comparison`**. La prima verrà utilizzata per creare un grafico che mostra l'andamento di una singola variabile rispetto a un asse temporale, con la possibilità di personalizzare etichette e titoli. La seconda funzione, invece, permette di confrontare visivamente due serie di dati, tracciando due curve differenti sullo stesso grafico per evidenziare somiglianze o differenze. Questa funzione include anche la possibilità di applicare una scala logaritmica ai dati, ci sarà particolarmente utile per rappresentare variabili con ampie variazioni di ordine di grandezza, come le pressioni.

Alla fine del codice viene utilizzata la classe **`TvacTest`** e le funzioni precedentemente definite per iniziare l'analisi e la visualizzazione dei dati raccolti durante il test. La sequenza di operazioni inizia con la creazione di un'istanza della classe, chiamata **`tvac`**. Successivamente, i dati vengono caricati nella variabile data tramite il metodo **`create_columns`**, che organizza le informazioni del file di log in un formato strutturato e facilmente accessibile.

Il metodo **`trend`** viene impiegato per calcolare le variazioni nella colonna relativa alla temperatura dello shroud. Questo metodo analizza l'andamento dei dati per determinare se la temperatura è in aumento, in diminuzione o stabile tra un valore e l'altro. Il risultato viene salvato nella variabile **`trend`**, mentre la variabile **`date_time`** contiene i dati temporali corrispondenti, anch'essi elaborati con il metodo **`trend`**. Con questi dati, la funzione **`plot_data`** crea un grafico che mostra il trend della temperatura dello shroud rispetto al tempo.
A questo punto, i dati specifici di interesse vengono estratti utilizzando il metodo **`create_array`**. In particolare, la temperatura dello shroud (T Shroud) e il relativo setpoint (T Shroud Setpoint) vengono recuperati e salvati rispettivamente nelle variabili **`t_shroud`** e **`t_shroud_setpoint`**. Questi due insiemi di dati vengono poi confrontati visivamente tramite la funzione comparison, che crea un grafico sovrapposto.

Infine, il codice analizza le pressioni all'interno della camera, misurate da due sensori distinti: il sensore "Full Range 1 chamber-ITR90" e il sensore "P Chamber". I valori corrispondenti vengono estratti tramite il metodo **`create_array`** e salvati nelle variabili **`P_Full_Range_1_chamber_ITR90`** e **`P_Chamber`**. Anche in questo caso, la funzione **`comparison`** viene utilizzata per creare un grafico che mette a confronto le letture dei due

sensori. Grazie all'opzione di scala logaritmica attivata con il parametro **`scale=True`**, il grafico riesce a rappresentare con precisione le variazioni su più ordini di grandezza.

Capitolo 3: Analisi dei Grafici

Il primo grafico presentato nella Figura 3-1 mostra il confronto tra due serie temporali relative a pressioni nella camera, una misurata da un sensore chiamato "P Full Range 1 chamber-ITR90" (in blu) e l'altra da "P_Chamber" (in rosso).
La scala logaritmica dell'asse y ci indica che i dati coprono un ampio intervallo di valori di pressione. Entrambe le curve mostrano comportamenti simili, con variazioni significative verso la data del 6 agosto.
Dopo il picco la pressione scende rapidamente e si stabilizza a valori più bassi. La correlazione tra i due segnali indica che entrambi i sensori registrano variazioni compatibili.

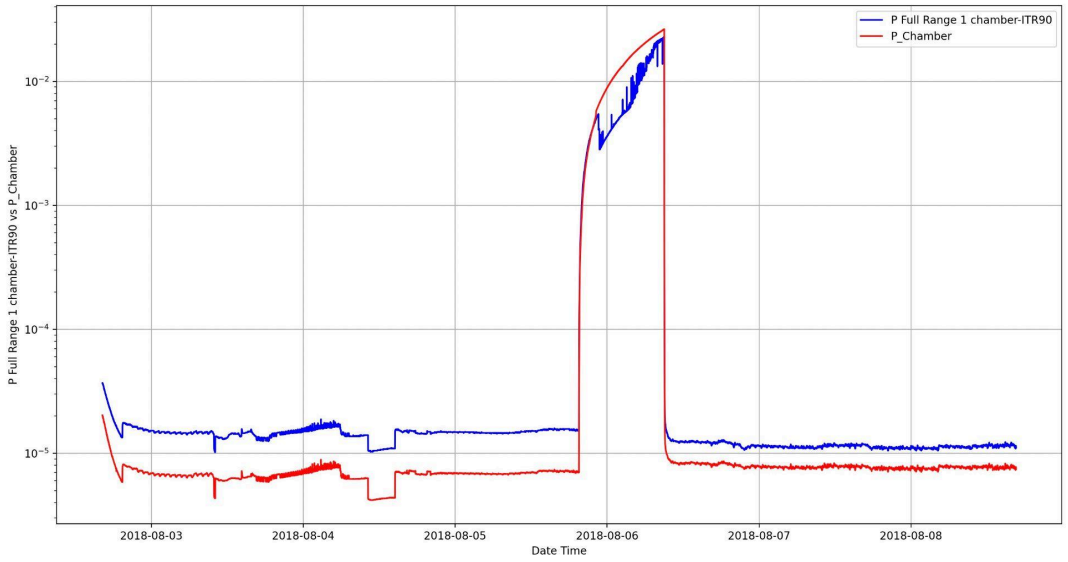

Figura 3-1

Il secondo grafico (Figura 3-2) mostra un confronto tra la temperatura registrata dello shroud (linea blu) e il setpoint associato (linea rossa) nel corso del tempo. La linea rossa, che rappresenta il setpoint, evidenzia un andamento discreto con variazioni di temperatura in momenti specifici. La linea blu, che corrisponde alla temperatura reale dello shroud, segue l'andamento del setpoint con un certo ritardo e presenta delle fluttuazioni. Queste oscillazioni indicano che il sistema richiede del tempo per adattarsi alle variazioni di temperatura impostate.

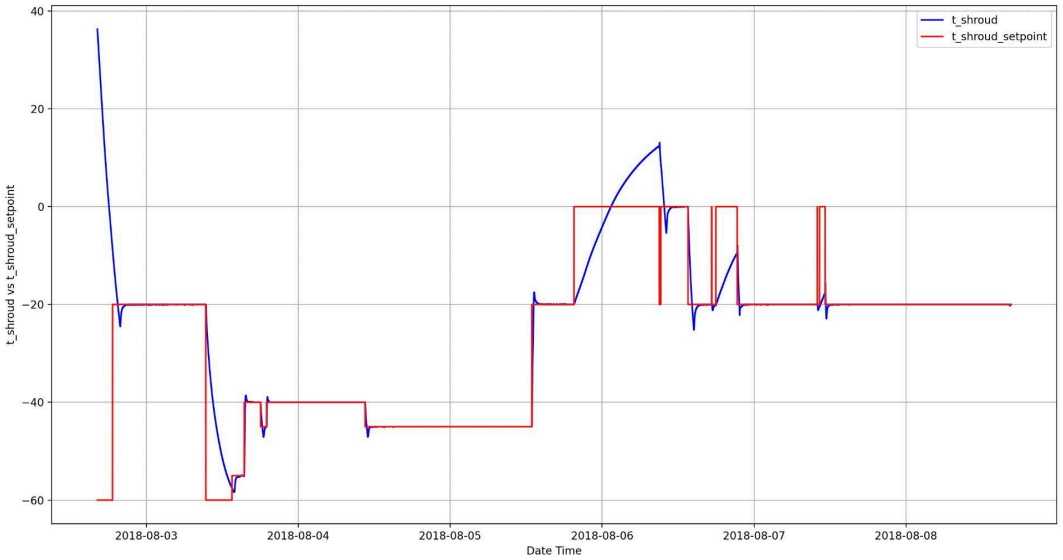

Figura 3-2

Il terzo grafico (Figura 3-3) rappresenta il trend della temperatura dello shroud nel tempo. I valori possono essere stabili (0), in aumento (+20) o in diminuzione (-20).Le frequenti variazioni nel grafico indicano che la temperatura dello shroud subisce continui aggiustamenti.Le sezioni con trend stabili (valori uguali a 0) indicano che il sistema ha raggiunto una condizione di equilibrio per brevi periodi (o che non viene superata la soglia impostata nel programma).grafico evidenzia una certa difficoltà del sistema nel mantenere la stabilità della temperatura, con oscillazioni frequenti.

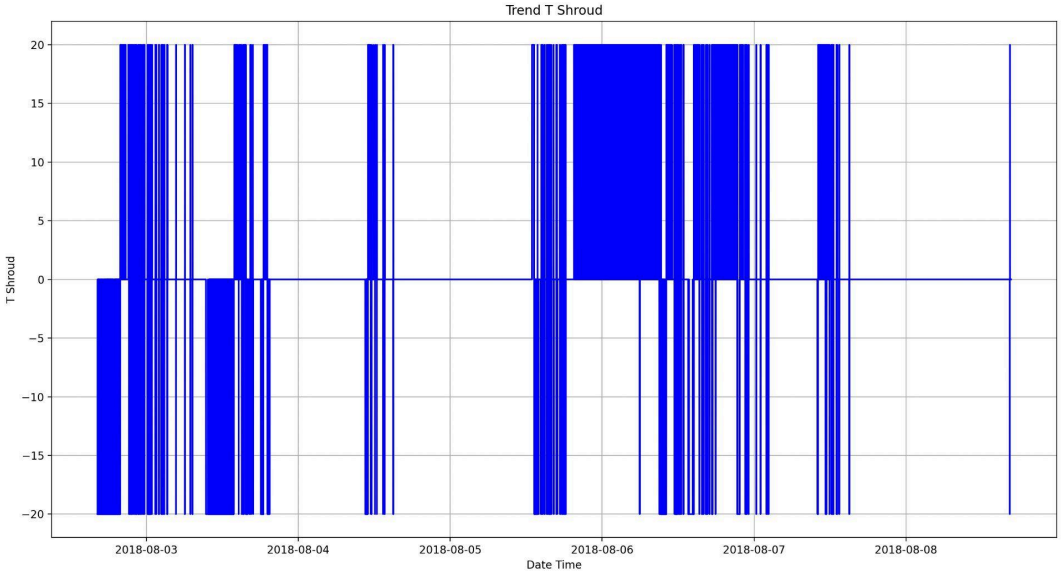

Figura 3-3

Conclusioni

L’esercitazione ha permesso di applicare le nostre conoscenze pregresse ed affinare tecniche di programmazione in Python per analizzare e visualizzare dati complessi provenienti da un sistema come il TVAC. Lo sviluppo di questo codice ha consolidato competenze fondamentali come la gestione di file, l’elaborazione di dati, e la generazione di grafici.
La scrittura del codice ci ha spinto ad adottare un approccio modulare e strutturato, che ci ha permesso una risoluzione efficiente dei problemi e delle richieste. In sintesi, questo lavoro ha rappresentato un’importante esperienza di apprendimento, evidenziando come strumenti informatici possano essere efficacemente utilizzati per analizzare dati provenienti dal mondo reale.

Bibliografia

1. NumPy - NumPy User Guide: https://numpy.org/doc/stable/user/index.html#user
   Documentazione ufficiale della libreria NumPy. (Accesso: Dicembre 2024)
2. Matplotlib: https://matplotlib.org/stable/users/index.html#
   Documentazione ufficiale di Matplotlib (Accesso: Dicembre 2024)
3. NumPy - GenfromTxt:
   https://numpy.org/doc/stable/reference/generated/numpy.genfromtxt.html
   Guida alla funzione genfromtxt (Accesso: Dicembre 2024)
4. Spiegazione di base TVAC: https://en.wikipedia.org/wiki/Thermal_vacuum_chamber
   Panoramica generale sul funzionamento del TVAC (Accesso: Dicembre 2024)