Sviluppo di un contatore elettrico intelligente

Stefano Antonio Labianca

22 Gennaio 2024
a.a 2023/2024

matricola: 758364 **email:** s.labianca10@studenti.uniba.it

Caso di Studio per l'esame di Ingegneria della Conoscenza

Tabella dei contenuti

Figura 1: Esempio di contatore differenziale

1 Introduzione

Nell'arco della nostra giornata, usiamo diversi dispositivi elettronici e, alle volte, anche per diverse ore della giornata o addirittura per tutto il giorno.

Per chi abita nelle zone di campagna, o in abitazioni singole, usare molti dispositivi elettronici contemporaneamente, specialmente se hanno alti consumi o possiedono una classe energetica bassa, fa scattare il salvavita.

1.1 Dispositivo salvavita

Il "salvavita", o più propriamente detto interruttore differenziale, è un dispositivo che arresta il flusso di energia elettrica dal contatore di un'abitazione, proteggendo persone e animali.

Questi interruttori, monitorano la differenza di corrente in entrata e in uscita dal dispositivo e, quando la differenza di corrente in entrata e in uscita supera una certa soglia, allora l'interruttore scatta togliendo l'alimentazione al circuito.

1.2 Obiettivo del progetto

Il progetto si pone l'obiettivo di sviluppare un programma in grado di svolgere i seguenti task:

1. Determinare da una lista di dispositivi, quali possono tenere accesi contemporaneamente senza che salti il salvavita.
2. Tenere traccia dei dispositivi elettronici e del loro consumo in Watt.
3. Ottenere tutti quei dispositivi che rispettano certi vincoli di consumo energetico.

2 Il progetto

2.1 Inizializzare il progetto

2.1.1 Scaricare da GitHub

Il primo passaggio è quello di clonare la repository cliccando al seguente link.

2.1.2 Impostare l'ambiente virtuale

Va creato successivamente un ambiente virtuale [2]. In questo modo l'interprete Python, le librerie e gli script installati al suo interno, saranno isolati dagli altri ambienti virtuali e da qualsiasi libreria installata sul proprio sistema.

Per creare l'ambiente virtuale, entrate nella cartella del progetto e digitate il seguente comando:

```
python -m venv .venv
```

Grazie a questo comando, verrà creata una cartella `/.venv` che conterrà tutto il necessario per lavorare con l'ambiente virtuale.

Una volta creato, è necessario attivare l'ambiente virtuale. Se siente in ambiente MacOS o Linux, digitate il comando

```
source .venv/bin/activate
```

Il file `activate` serve per "accendere" l'ambiente virtuale.

Se invece siete in ambiente Windows, allora posizionatevi prima dentro la cartella `./venv/Scripts/` per poi digitare uno dei seguenti comandi, in base al tipo di terminale in uso:

```
activate.bat // Se usi il CMD
.\Activate.ps1 // Se usi la PowerShell
```

Se invece si sta usando il GitBash, in ambiente Windows, allora il comando da appliace è il seguente:

```
source .venv/Scripts/activate
```

L'ambiente virtuale sarà attivato con successo quando sul vostro terminale avrete qualcosa di simile alla figura 2.

In caso di problemi nell'uso della PowerShell, è possibile andare alla sezione: Risolvere il problema di esecuzione con la PowerShell.

2.1.3 Avviare il progetto

Per avviare il progetto, bisogna tornare alla directory principale del progetto e installare le dipendenze con il comando:

```
pip install -r requirements.txt
```

Una volta installate, digitare il seguente comando per avviare il programma

```
python main.py
```

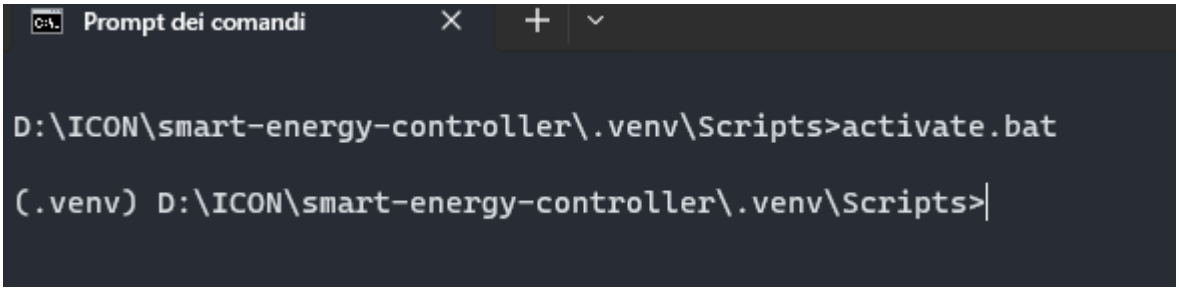
In caso di errore, vedere la sezione: Risolvere il problema di esecuzione con la PowerShell.

2.2 Struttura del progetto

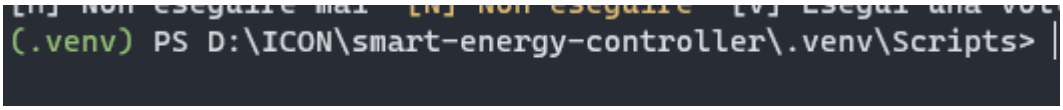
All'intero del progetto, possiamo trovare le seguenti cartelle:

- `/.vevn`: Cartella contenente tutto il necessario per lavorare con l'ambientevirtuale.
- `/appliance`: Qui è possibile trovare la classe Appliance, che definisce un elettrodomestico, insieme ad una serie di metodi di suppporto, contenuti in `appliances_controller.py`

(a) Uso di GitBash



(b) Uso del CMD



(c) Uso della PowerShell

Figura 2: Risultato dell'attivazione dell'ambiente virtuale. In figura (a) abbiamo l'uso del GitBash, in figura (b) del CMD mentre infine, nella figura (c), abbiamo l'uso della PowerShell.

- `/cli`: Troviamo una classe che incapsula tutta la logica legata agli input e all'output del terminale
- `/csp_problem`: Questa cartella contiene tutti i file legati all'argomento del CSP.
  Infatti è possibile trovare la rappresentazione delle variabili e dei vincoli, fatta rispettivamente usando le classi Variable e Constraint.
  Inoltre è presente anche la classe CSP usata come wrapper per rappresentare un generico problema di questa categoria.
  Infine è presenta la cartella `/algorithm` che contiene le realizzazioni degli algoritmi DFS e GAC usati per risolvere i problemi legati al CSP.
- `/knowledge_base`: Contiene una classe usata per rappresentare il Sis-

tema Esperto realizzato.

- `/ontology`: Questa cartella contiene una classe che permette di manipolare l'ontologia contenuta all'intero del file `appliance_ontology.rdf`.
- `/test`: Contiene tutti quei file contenente vari test fatti al programma.
- `/utils`: Contiene file di utilità che facilitano alcune operazioni interne al programma. Per esempio, il file `pagination.py` viene usato per impaginare l'output del programma.

2.3 Scelte progettuali

Il progetto è stato svolto usando la versione `3.11.5` di Python in quanto offre diversi miglioramenti e un supporto maggiore alla tipizzazione delle variabili e delle constanti. Usare i tipi, infatti, si è rivelato molto utile per rendere la codebase più resiliente, evitando di assegnare, erroneamente, valori con tipo non corretto per una variabile.

L'uso del design pattern del Singleton si è rivelato utile in quando permette l'uso di una sola istanza globale della classe `ApplianceOntology`. Questa classe è stata usata per rappresentare le informazioni dell'ontologia usata e di manipolarla, tramite operazioni di lettura e scrittura.

L'istanza creata, infatti, viene usata in più parti del programma e, creare istanze differenti fra i vari moduli, rischieremmo di avere istanze non aggiornate ai cambiamenti fatti in altre parti del programma.

Infine per l'argomento del CSP [1], sono state usate delle versioni leggermente modificate degli algoritmi DFS e GAC fornite dal libro di testo AIPython, in quando si è scelto di adattarle al problema imposto dal progetto. Non solo, lo stesso vale per le classi `Variable` e `Contraint` dove sono stati presi in considerazione solamente le funzionalità essenziali.

3 Argomenti trattati

4 Troubleshooting

4.1 Risolvere il problema di esecuzione con la PowerShell

In caso non si riesca ad eseguire con la PowerShell [3], l'attivazione dell'ambiente virtuale, provate a svolgere i seguenti passi.

Aprire innanzitutto la PowerShell come amministratore nella cartella del progetto. Una volta aperta la PowerShell, digitare il comando:

```
Get-ExecutionPolicy
```

Grazie a questo comando, possiamo sapere quale execution policy è impostata per la PowerShell. In figura 3, è mostrato una possibile execution policy impostata nella PowerShell.

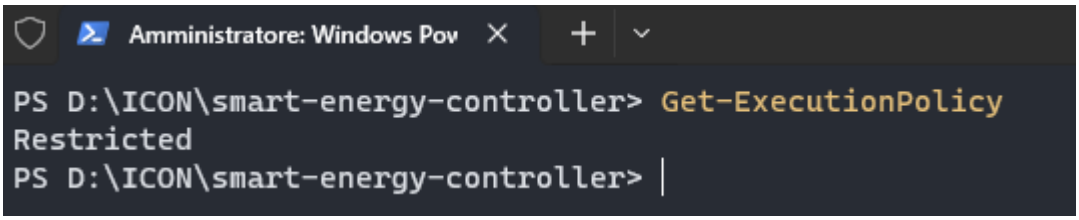


Figura 3: Possibile execution policy

Per permettere l'esecuzione degli script `.ps1`, allora bisogna impostare su `AllSigned` la execution policy.
Per farlo si usa il comando:

```
Set-ExecutionPolicy -ExecutionPolicy AllSigned
```

Una volta eseguito questo comando, andare nella cartella `/.venv/Scripts/` e digitare il comando:

```
.\Activate.ps1
```

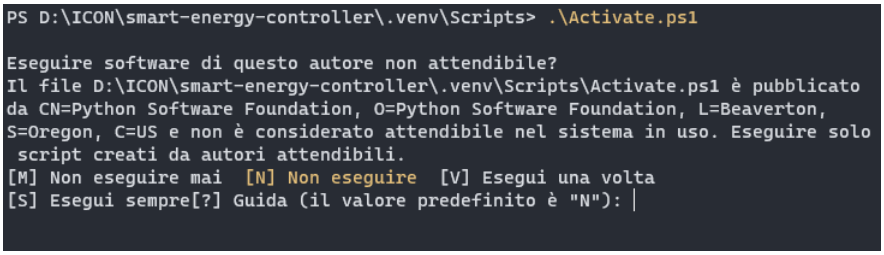


Figura 4: Messaggio di conferma

Apparirà sul terminale il seguente output:

Per eseguire lo script di attivazione, allora inserite `V` e premete invio.

Una volta che avete finito l'esecuzione del programma, potete anche reimpostare la execution policy al suo stato orinario, usando il comando:

```
Set-ExecutionPolicy -ExecutionPolicy <PolicyNamePrecedente>
```

4.2 Errore di esecuzione del programma Python

E' possibile che, durante l'esecuzione del programma, possa apparire il seguente messaggio di errore [4]:

```
Utente@DESKTOP-4KVIMPN MINGW64 /d/ICON/smart-energy-controller (main)
$ python main.py
Traceback (most recent call last):
  File "D:\ICON\smart-energy-controller\main.py", line 1, in <module>
    from knowledge_base.expert_system import run_expert_system
  File "D:\ICON\smart-energy-controller\knowledge_base\expert_system.py", line 1, in <module>
    from experta import DefFacts, Fact, KnowledgeEngine, Rule
  File "D:\ICON\smart-energy-controller\.venv\Lib\site-packages\experta\__init__.py", line 5, in <module>
    from .engine import KnowledgeEngine
  File "D:\ICON\smart-energy-controller\.venv\Lib\site-packages\experta\engine.py", line 13, in <module>
    from experta.fact import InitialFact
  File "D:\ICON\smart-energy-controller\.venv\Lib\site-packages\experta\fact.py", line 9, in <module>
    from experta.utils import freeze, unfreeze
  File "D:\ICON\smart-energy-controller\.venv\Lib\site-packages\experta\utils.py", line 4, in <module>
    from frozendict import frozendict
  File "D:\ICON\smart-energy-controller\.venv\Lib\site-packages\frozendict\__init__.py", line 16, in <module>
    class frozendict(collections.Mapping):
                     ^^^^^^^^^^^^^^^^^^^
AttributeError: module 'collections' has no attribute 'Mapping'
(.venv)
```

Figura 5: Errore di esecuzione

Questo errore è dovuto ad una versione datata della libreria `frozendict` usata come dipendenza della libreria `experta`.

Fare l'upgrade della libreria `frozendict` alla versione più recente, andrebbe a creare dei conflitti di dipendenza tra le due versioni della librerie.

Per risolvere questo problema, bisogna andare nella cartella `/.venv/Lib/site-packages/frozendict` e aprire il file `__init__.py`.
Una volta aperto, bisogna cambiare la seguente linea di codice:

```
class frozendict(collections.Mapping):
    ...
```

Nella seguente:

```
class frozendict(collections.abc.Mapping):
    ...
```

5 Riferimenti Bibliografici

[1] AIPython (2023): Ragionamento con i vincoli, capitolo 4 pagina 69. .

[2] freeCodeCamp (2022): Creazione di un ambiente virtuale in Python

[3] PowerShell Documentation (2022): Documentazione sul funzionamento delle execution policy della PowerShell Windows

[4] Attribute Error (2023): Risoluzione problema legato alle dipendenze datate