



Splunk IoT

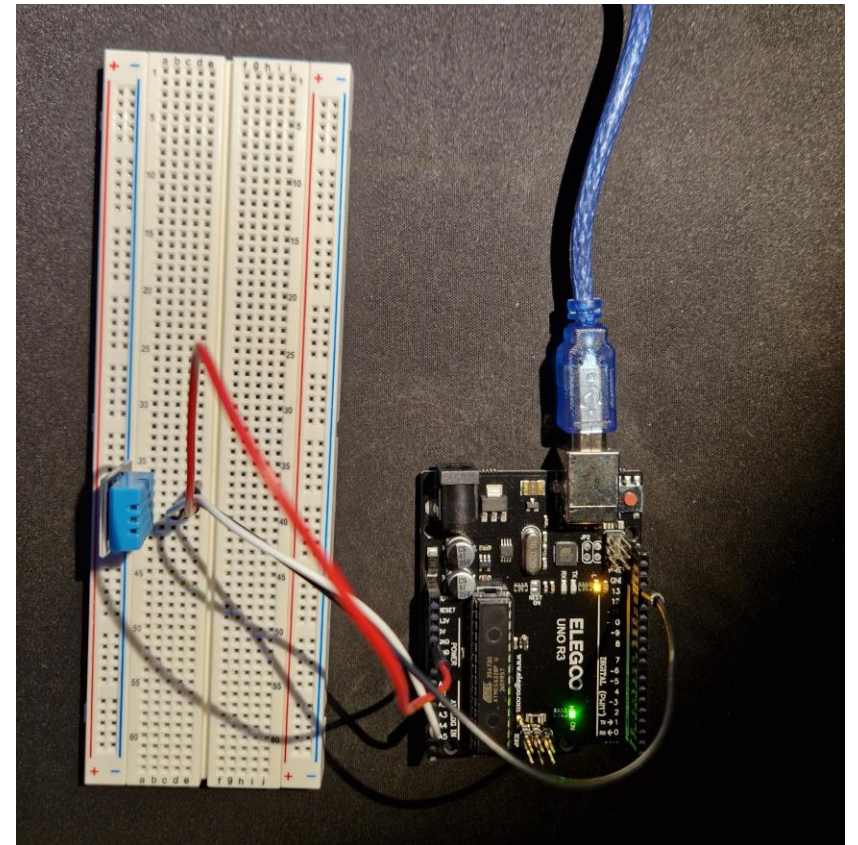
Descrizione del progetto

L'obiettivo è quello di dimostrare le potenzialità di un **SIEM** (in questo caso **Splunk**) nell'attività di monitoraggio di un dispositivo **IoT**.

Grazie a **Splunk** abbiamo la possibilità di isolare e riorganizzare i dati in tabelle o in strumenti grafici che ci aiutano a sintetizzare le informazioni e visualizzarle in maniera intuitiva.

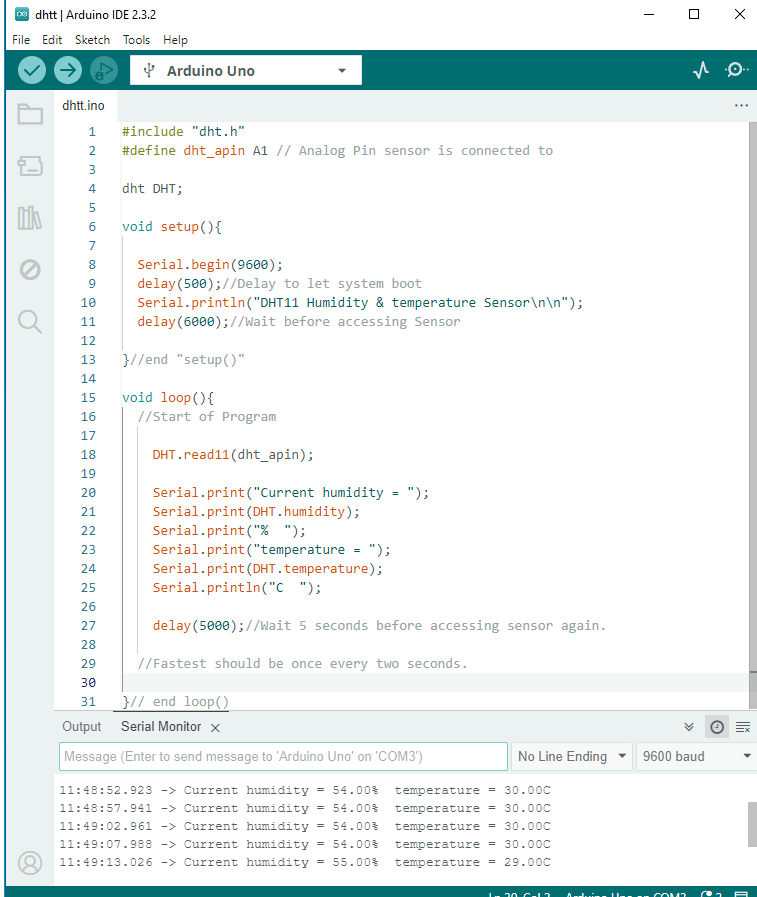
Il dispositivo IoT

Come dispositivo **IoT** ho scelto un sensore **DHT-11** che rileva temperatura e umidità ambientale, tramite una scheda **Arduino** e una breadboard posso comunicare i dati rilevati al pc.



Rilevamento e trasmissione dati

Al fine di verificare il corretto funzionamento del sensore ho prima provato a rilevare i dati sull'Ide di Arduino.



The screenshot shows the Arduino IDE 2.3.2 interface. The main editor displays a sketch named 'dht.ino' with the following code:

```
1 #include "dht.h"
2 #define dht_apin A1 // Analog Pin sensor is connected to
3
4 dht DHT;
5
6 void setup(){
7
8   Serial.begin(9600);
9   delay(500); // Delay to let system boot
10  Serial.println("DHT11 Humidity & temperature Sensor\n\n");
11  delay(6000); // Wait before accessing Sensor
12
13 } // end "setup()"
14
15 void loop(){
16   // Start of Program
17
18   DHT.read11(dht_apin);
19
20   Serial.print("Current humidity = ");
21   Serial.print(DHT.humidity);
22   Serial.print(" % ");
23   Serial.print("temperature = ");
24   Serial.print(DHT.temperature);
25   Serial.println("C ");
26
27   delay(5000); // Wait 5 seconds before accessing sensor again.
28
29   // Fastest should be once every two seconds.
30
31 } // end loop()
```

The bottom of the IDE shows the 'Serial Monitor' window, which is set to 'No Line Ending' and '9600 baud'. It displays the following output:

```
11:48:52.923 -> Current humidity = 54.00% temperature = 30.00C
11:48:57.941 -> Current humidity = 54.00% temperature = 30.00C
11:49:02.961 -> Current humidity = 54.00% temperature = 30.00C
11:49:07.988 -> Current humidity = 54.00% temperature = 30.00C
11:49:13.026 -> Current humidity = 55.00% temperature = 29.00C
```

The status bar at the bottom indicates 'Ln 30, Col 2 Arduino Uno on COM3'.

Rilevamento e trasmissione dati

Fatto ciò il problema era capire come trasmettere i dati su **Splunk**, tra le varie feature a disposizione della piattaforma c'è l'HEC (HTTP Event Collector) , un endpoint HTTP che consente agli utenti di inviare eventi e dati in tempo reale a **Splunk Enterprise** o **Splunk Cloud**.

HEC supporta la ricezione di eventi tramite richieste HTTP POST.

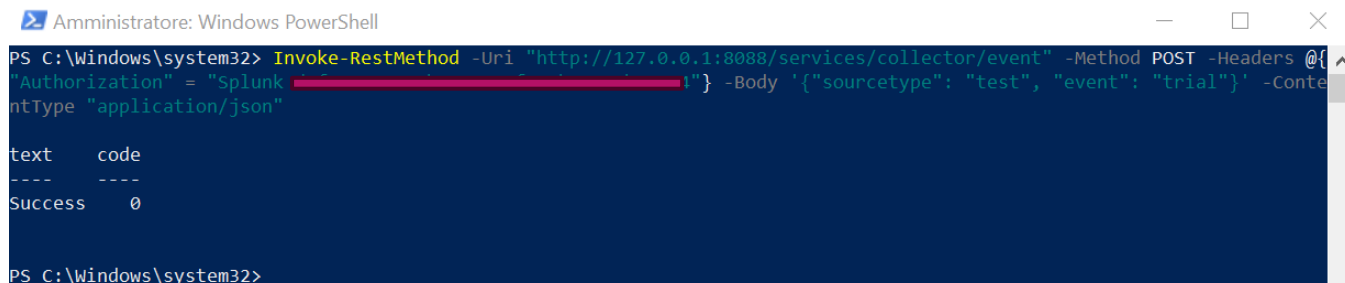
Prima di scrivere un programma che prendesse i dati dalla porta a cui era collegato il sensore ho testato l'endpoint inviando una richiesta HTTP POST «di prova» con Powershell come spiegato nella slide successiva, ma configurando prima l'endpoint su **Splunk** generando un token per connettermi effettuando un'autenticazione.

The screenshot shows the Splunk Enterprise web interface. At the top, the navigation bar includes the Splunk logo, user role (Administrator), and various menu items like Messaggi, Impostazioni, Attività, and Guida. A search bar is also present. The main content area is titled 'Raccolta eventi HTTP' (HTTP Event Collector). Below the title, there's a sub-header 'Input dati » Raccolta eventi HTTP'. A table lists the configured HEC endpoints. In this case, there is one entry named 'DHT temp' with a token value (redacted with a black box). The table also shows the source type as 'main' and the status as 'Abilitato' (Enabled).

Nome	Azioni	Valore token	Source type	Indice	Stato
DHT temp	Modifica Disabilita Elimina	[Redacted]	main	main	Abilitato

Rilevamento e trasmissione dati

- 1. Invoke-RestMethod:** Questo è un cmdlet di PowerShell che viene utilizzato per eseguire richieste HTTP come POST, GET, PUT, DELETE, ecc.
- 2. -Uri "http://127.0.0.1:8088/services/collector/event":** Questo è il parametro che specifica l'URL a cui verrà inviata la richiesta HTTP. Nel nostro caso, stiamo inviando i dati all'endpoint Splunk HEC (HTTP Event Collector) all'indirizzo specificato.
- 3. -Method POST:** Questo parametro specifica che vogliamo inviare una richiesta di tipo POST all'URL specificato. In una richiesta POST, stiamo inviando dati al server.
- 4. -Headers @{"Authorization" = "Splunk[REDACTED]"}:** Questo parametro specifica gli header della richiesta HTTP. Qui stiamo includendo l'header di autorizzazione necessario per autenticarci con Splunk HEC. L'autorizzazione viene fornita utilizzando il token Splunk HEC.
- 5. -Body '{"sourcetype": "test", "event": "trial"}':** Questo parametro specifica il corpo della richiesta HTTP. Qui stiamo includendo i dati che vogliamo inviare al server. Il corpo della richiesta è un oggetto JSON che include i campi "sourcetype" e "event" con i valori corrispondenti.
- 6. -ContentType "application/json":** Questo parametro specifica il tipo di contenuto del corpo della richiesta HTTP. In questo caso, stiamo indicando che il corpo della richiesta è in formato JSON.



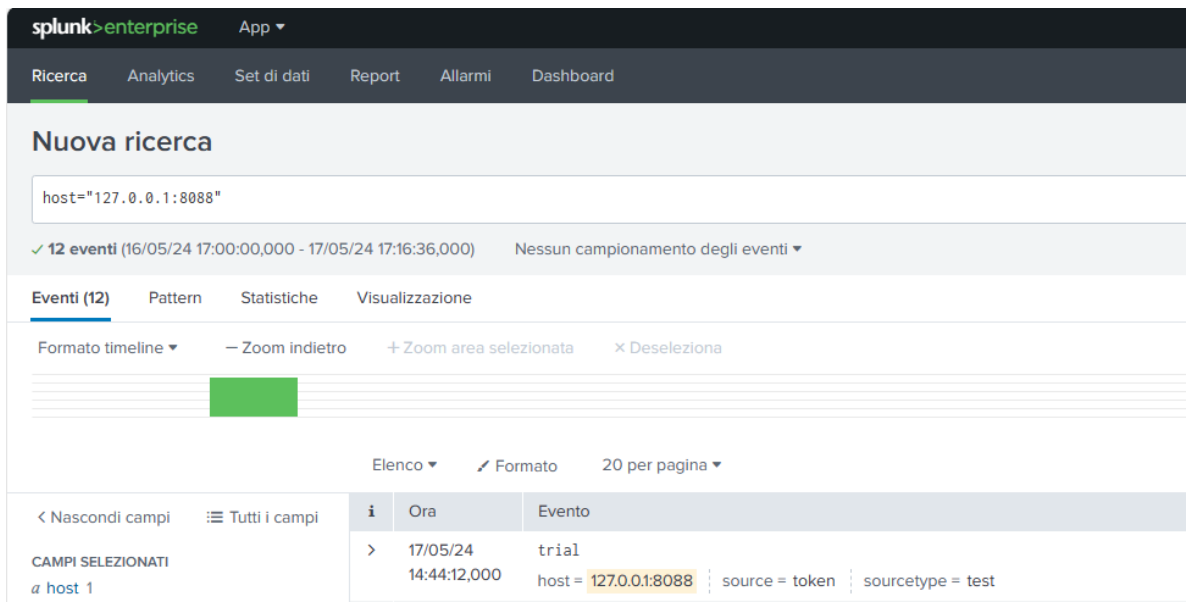
```
Amministratore: Windows PowerShell
PS C:\Windows\system32> Invoke-RestMethod -Uri "http://127.0.0.1:8088/services/collector/event" -Method POST -Headers @{"Authorization" = "Splunk[REDACTED]"} -Body '{"sourcetype": "test", "event": "trial"}' -ContentType "application/json"

text      code
----      -
Success    0

PS C:\Windows\system32>
```

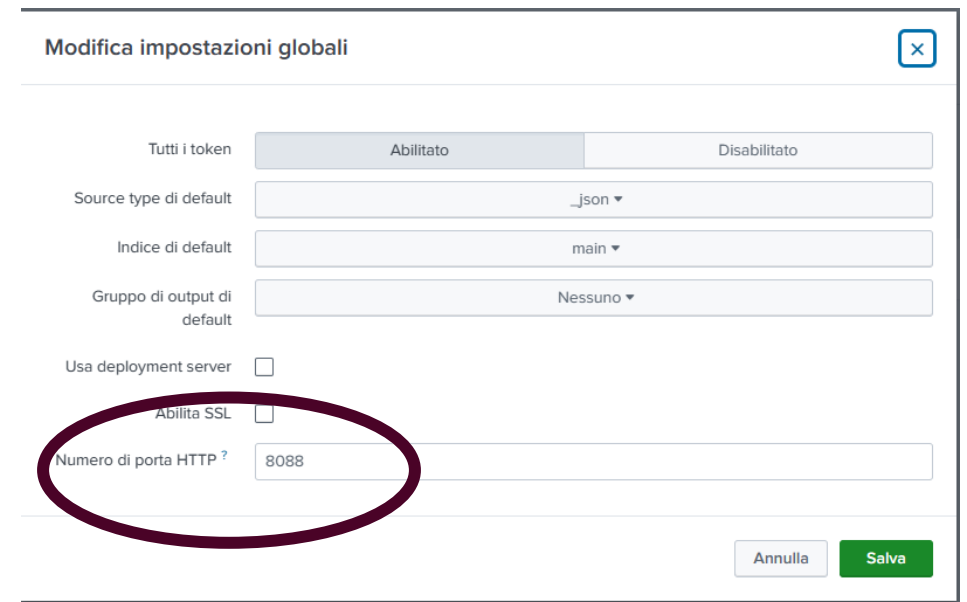
Rilevamento e trasmissione dati

Inserendo l'indirizzo e la porta che corrispondono all'endpoint nel campo di ricerca su **Splunk** possiamo osservare la ricezione dei dati inviati (la porta 8088 risulta nelle impostazioni globali del collector).



The screenshot shows the Splunk search interface. At the top, the navigation bar includes 'splunk>enterprise', 'App', and tabs for 'Ricerca', 'Analytics', 'Set di dati', 'Report', 'Allarmi', and 'Dashboard'. The 'Ricerca' tab is active. Below the navigation bar, the 'Nuova ricerca' section shows a search query 'host="127.0.0.1:8088"'. The results show 12 events from 16/05/24 17:00:00,000 to 17/05/24 17:16:36,000. The 'Eventi (12)' tab is selected, and a timeline visualization is shown. Below the timeline, the 'Elenco' tab is selected, showing a table of events. The first event is from 17/05/24 at 14:44:12,000, with the source 'host = 127.0.0.1:8088' highlighted in yellow. The source type is 'token' and the sourcetype is 'test'.

i	Ora	Evento
>	17/05/24 14:44:12,000	trial host = 127.0.0.1:8088 source = token sourcetype = test



The screenshot shows the 'Modifica impostazioni globali' dialog box. The settings are as follows:

- Tutti i token: Abilitato
- Source type di default: _json
- Indice di default: main
- Gruppo di output di default: Nessuno
- Usa deployment server: ☐
- Abilita SSL: ☐
- Numero di porta HTTP: 8088

The 'Numero di porta HTTP' field is circled in red. At the bottom right, there are 'Annulla' and 'Salva' buttons.

Rilevamento e trasmissione dati

Dato che la richiesta con **PowerShell** funziona ho pensato semplicemente di reitenerla e modificarla in modo che nel json venissero riportati i dati relativi alla temperatura e l'umidità rilevati dal sensore (importando una libreria in python che mi permettesse di fare quanto fatto su **Arduino**). Ho inoltre modificato l'endpoint puntando all'indirizzo «raw» che mi ha permesso di ricevere i dati in maniera meno vincolata al formato. Di seguito la schermata di **Powershell** dove avvio il programma, il codice del programma in Python e la dashboard di **Splunk** dove ricevo i dati che posso riorganizzare in base alle informazioni che devo ricavare.


```
{"sourcetype": "test", "event": "trial", "humidity": 50.0, "temperature": 29.0}
```

Comando PowerShell:

```
Invoke-RestMethod -Uri "http://127.0.0.1:8088/services/collector/raw" -Method POST -Headers @{"Authorization" = [REDACTED] -Body "{\"sourcetype\": \"test\", \"event\": \"trial\", \"humidity\": 50.0, \"temperature\": 29.0}" -ContentType "application/json"
```

```
text    code
```

```
-----
```

```
Success 0
```

JSON da inviare a Splunk:

```
{"sourcetype": "test", "event": "trial", "humidity": 49.0, "temperature": 29.0}
```

Comando PowerShell:

```
Invoke-RestMethod -Uri "http://127.0.0.1:8088/services/collector/raw" -Method POST -Headers @{"Authorization" = [REDACTED] -Body "{\"sourcetype\": \"test\", \"event\": \"trial\", \"humidity\": 49.0, \"temperature\": 29.0}" -ContentType "application/json"
```

```
text    code
```

```
-----
```

```
Success 0
```

JSON da inviare a Splunk:

```
{"sourcetype": "test", "event": "trial", "humidity": 49.0, "temperature": 29.0}
```

Comando PowerShell:

```
Invoke-RestMethod -Uri "http://127.0.0.1:8088/services/collector/raw" -Method POST -Headers @{"Authorization" = [REDACTED] -Body "{\"sourcetype\": \"test\", \"event\": \"trial\", \"humidity\": 49.0, \"temperature\": 29.0}" -ContentType "application/json"
```

```
text    code
```

```
-----
```

```
Success 0
```

JSON da inviare a Splunk:

```
{"sourcetype": "test", "event": "trial", "humidity": 48.0, "temperature": 29.0}
```

Comando PowerShell:

```
Invoke-RestMethod -Uri "http://127.0.0.1:8088/services/collector/raw" -Method POST -Headers @{"Authorization" = [REDACTED] -Body "{\"sourcetype\": \"test\", \"event\": \"trial\", \"humidity\": 48.0, \"temperature\": 29.0}" -ContentType "application/json"
```

```
text    code
```

```
-----
```

```
Success 0
```

```
Welcome  def.py 1 X
C: > Users > royve > Desktop > def.py > ...
1 import serial
2 import json
3 import subprocess
4
5 # Configurazione di Splunk HEC
6 hec_url = "http://127.0.0.1:8088/services/collector/raw"
7 hec_token = 
8
9 # Imposta la porta seriale e la velocità di comunicazione
10 ser = serial.Serial('COM3', 9600) # Assicurati di sostituire 'COM3' con la porta seriale corretta
11
12 try:
13     while True:
14         # Leggi la risposta dalla porta seriale
15         response = ser.readline().decode().strip()
16
17         # Se la riga inizia con "Current humidity"
18         if response.startswith("Current humidity"):
19             # Estrai i valori di umidità e temperatura come stringhe
20             humidity = response.split("humidity = ")[1].split("%")[0]
21             temperature = response.split("temperature = ")[1].split("C")[0]
22
23             # Crea un dizionario con i dati
24             data = {
25                 "sourcetype": "test",
26                 "event": "trial",
27                 "humidity": float(humidity),
28                 "temperature": float(temperature)
29             }
30
31             # Converti il dizionario in formato JSON
32             json_data = json.dumps(data)
33
34             # Stampa il JSON prima di inviarlo
35             print("JSON da inviare a Splunk:")
36             print(json_data)
37
38             # Formatta il JSON in modo che sia compatibile con PowerShell
39             formatted_json_data = json_data.replace("'", '"')
40
41             # Costruisci il comando PowerShell con il JSON correttamente formattato
42             powershell_command = f'Invoke-RestMethod -Uri "{hec_url}" -Method POST -Headers @{{"Authorization" = "Splunk {hec_token}"}} -Body "{formatted_json_data}" -ContentType "application/json"'
43
44             # Stampa il comando PowerShell
45             print("Comando PowerShell:")
46             print(powershell_command)
47
48             # Esegui la richiesta HTTP utilizzando Invoke-RestMethod di PowerShell
49             subprocess.run(["powershell", "-Command", powershell_command], shell=True)
```

Nuova ricerca

Salva come

Crea vista tabella

Chiudi

host="127.0.0.1:8088"

Ultime 24 ore

✓ 16 eventi (16/05/24 17:00:00,000 - 17/05/24 17:45:51,000) Nessun campionamento degli eventi

Processo

Modalità dettagliata

Eventi (16)

Pattern

Statistiche

Visualizzazione

Formato timeline

Zoom indietro

Zoom area selezionata

Deseleziona

1 ora per colonna

Elenco

Formato

20 per pagina

< Nascondi campi

Tutti i campi

CAMPI SELEZIONATI

a host 1

a source 1

a sourcetype 2

CAMPI INTERESSANTI

a event 1

a extracted_sourcetype 1

humidity 4

a index 1

linecount 1

a punct 2

a splunk_server 1

temperature 1

a timestamp 1

+ Estrai nuovi campi

i	Ora	Evento
>	17/05/24 17:38:15,000	<div>{ [-]</div> <div>event: trial</div> <div>humidity: 52.00</div> <div>sourcetype: test</div> <div>temperature: 28.00</div> <div>}</div> <div>Mostra come testo non elaborato</div> <div>host = 127.0.0.1:8088 source = token sourcetype = _json</div>
>	17/05/24 17:38:10,000	<div>{ [-]</div> <div>event: trial</div> <div>humidity: 52.00</div> <div>sourcetype: test</div> <div>temperature: 28.00</div> <div>}</div> <div>Mostra come testo non elaborato</div> <div>host = 127.0.0.1:8088 source = token sourcetype = _json</div>
>	17/05/24 17:38:05,000	<div>{ [-]</div> <div>event: trial</div> <div>humidity: 52.00</div> <div>sourcetype: test</div> <div>temperature: 28.00</div> <div>}</div> <div>Mostra come testo non elaborato</div> <div>host = 127.0.0.1:8088 source = token sourcetype = _json</div>

Conclusioni

Con un sistema del genere ho la possibilità di monitorare nel tempo i dati di un dispositivo **IoT** da remoto ed avere una visualizzazione sintetica, grafica e intuitiva dei dati tramite le funzioni di **Splunk Enterprise** come nella slide seguente.

La ricerca:

host="127.0.0.1:8088" humidity=* | timechart span=10s avg(humidity)

calcola la media dell'umidità per ogni 10 secondi e utilizza il timestamp per tracciare l'andamento temporale.

Nuova ricerca

Salva come

Crea vista tabella

Chiudi

host="127.0.0.1:8088" humidity=* | timechart span=10s avg(humidity)

Sempre (tempo reale)

Q

14 di 14 eventi corrispondenti

Nessun campionamento degli eventi

Processo

⏸

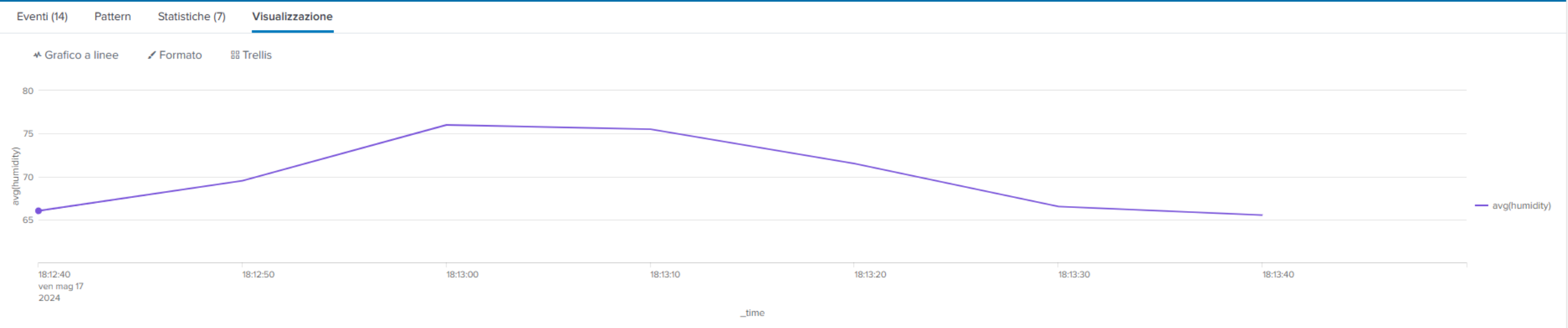
■

↗

🖨

⬇

Modalità dettagliata



_time	avg(humidity)
2024-05-17 18:12:40	66
2024-05-17 18:12:50	69.5
2024-05-17 18:13:00	76
2024-05-17 18:13:10	75.5
2024-05-17 18:13:20	71.5
2024-05-17 18:13:30	66.5
2024-05-17 18:13:40	65.5