



Astro Pi - Concorso ESA

Elia Ernesto Stellin - Programmatore Junior @ Emme informatica Srl

@PyDataVenice #15 #Meetup #PyData

Elia Ernesto Stellin

- Studente di Informatica @ UniPD
- Programmatore Junior @ Emme Informatica Srl
- Partecipato al concorso Astro-Pi “Mission Space Lab” dell’ESA

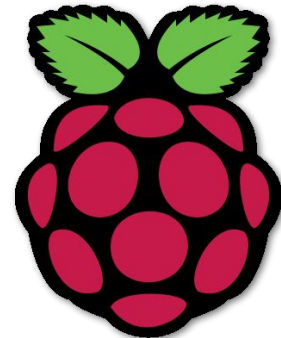
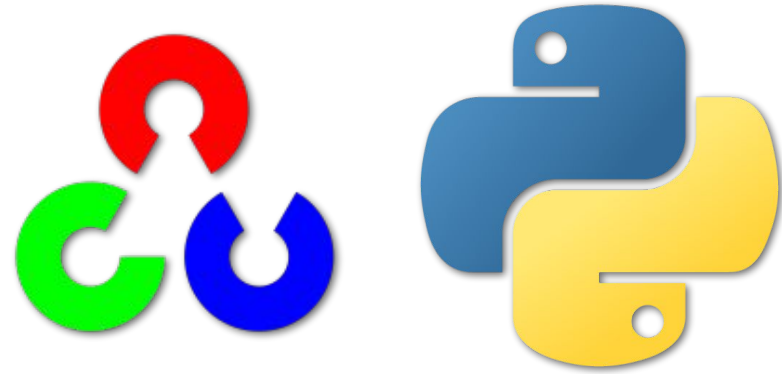


elia-ernesto-stellin



Obiettivi del talk

- 1) Mostrare use case di **OpenCV** con **Python** per un progetto reale
- 2) Mostrare use case possibile per esperimenti col **Raspberry Pi**



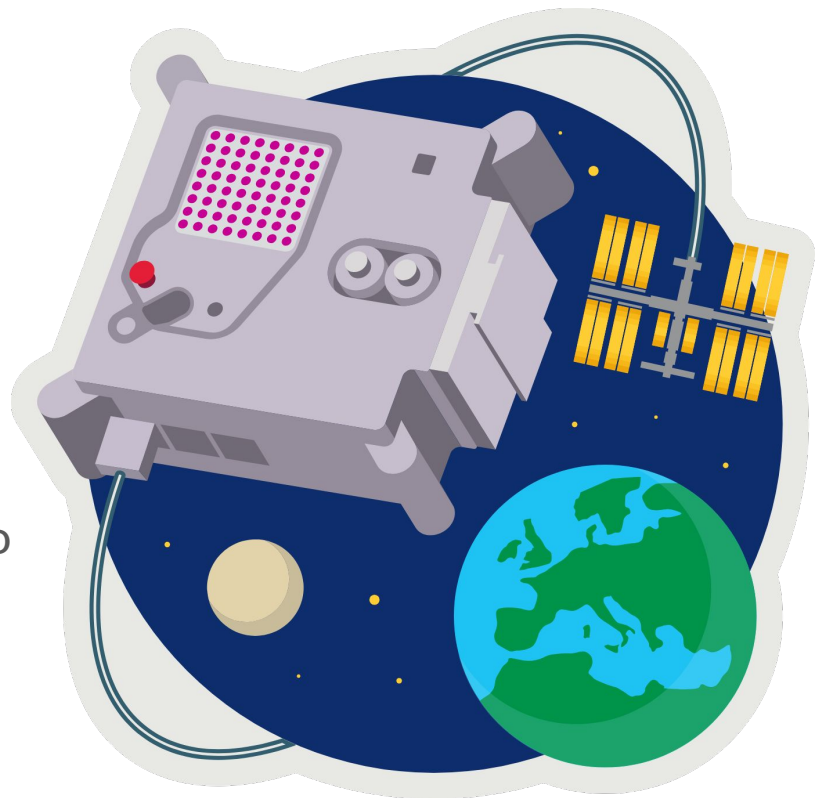
Contenuti


- Overview del progetto Astro-Pi dell'ESA
- Fase 1: Ideazione
- Fase 2: Creazione del programma
- Fase 3: Flight Status
- Fase 4: Analisi dati

Overview del Concorso

Astro-Pi Mission Space Lab (2022-23)

- Concorso dell'ESA in 4 fasi
- Progettare con **Python** un esperimento, che verrà eseguito su un **Raspberry Pi** sulla ISS
- Esperimento da **3 ore** su Terra o spazio
- Molti **sensori** da usare
- Alla fine, **analisi dati finale** sui dati ricevuti dallo spazio





Fasi 1-2: Ideazione e creazione dell'esperimento

Settembre 2022 – Febbraio 2023

- Esperimento scelto
- Idea di base
- Microgravità
- Creazione del programma
- Sensori
- Human Detection

Fase 1: Ideazione dell'esperimento

Esperimento scelto: *Spazio*

- Idea di base: rilevare **microgravità nella ISS** tra Astro-Pi e gli astronauti
- Come?
 - **Accelerometro**: rilevare accelerazione di microgravità
 - **Sensore IR**: rilevare possibile presenza umana
 - **Fotocamera + Human Detection**: rilevare umani con certezza

Cos'è la microgravità?

- Accelerazione di gravità molto piccola
- Legge della **gravitazione universale**:
due corpi si attraggono secondo
[Legge di Newton](#)
- Si può rilevare questa forza con
accelerometro?
- Valore atteso: $10^{-8} \sim 10^{-10} \text{ m/s}^2$
 - Dai test, valori con 10^{-15}
 - 1 / 100 miliardi della gravità terrestre!)

$$F = G \frac{m_1 m_2}{d^2}$$

$$G = 6.6743 \cdot 10^{-11} \frac{\text{Nm}^2}{\text{kg}}$$

$$m_1 = 0.05 \text{ kg} \text{ (AstroPi)}$$

$$m_2 = 70 \text{ kg} \text{ (Essere umano)}$$

$$d = 0.5 \text{ m} \text{ (Distanza tra i due)}$$

$$\Rightarrow F = 9.34 \cdot 10^{-10} \frac{\text{m}}{\text{s}^2}$$

VALORE MOLTO PICCOLO!

Fase 2: Creazione del programma

Vincoli / requisiti del concorso:

- Programma in **Python**
- Tempo massimo di **3 ore**
- Spazio limitato (max **3 GB**)
- **Non si possono salvare immagini dell'interno della ISS**
- Si possono usare solo certe librerie Python



Thomas Pesquet a bordo della ISS con due Astro-Pi

Fase 2: Creazione del programma

Idea generale:

1. Rileva dati dai sensori molto spesso
2. Salva dati dei sensori su CSV
3. Ogni tanto scatta foto con telecamera (*ogni 60 iterazioni*)
4. Analizza foto per riconoscere essere umano
5. Salva i risultati dell'analisi

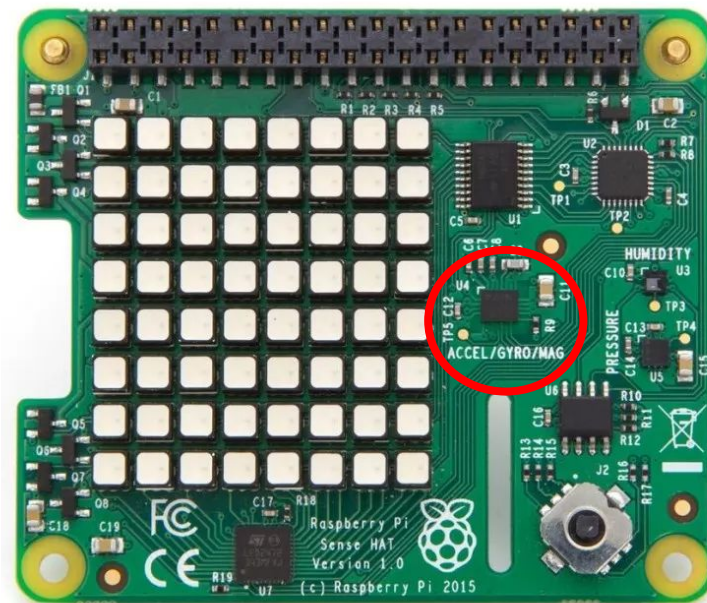
Rilevare dati dai sensori

Ogni 20 ms, il programma salva i dati di tutti sensori (con la libreria `sense_hat.SenseHat`), tra cui:

- **Accelerometro**
- Giroscopio, magnetometro
- Temperatura, umidità, pressione

Abbiamo anche utilizzato:

- **Sensore a IR** (gpiozero)
- **GPS** (orbit.ISS)



La scheda Pi Sense Hat

Riconoscere un essere umano

Ogni 1.2 s: il programma scatta una foto e la analizza usando **OpenCV**.

Moduli usati: **SVM + HOG descriptor**.

- **HOG** = *feature descriptor*
 - **Input**: immagine che rappresenta un soggetto
 - **Output**: informazioni su *feature* del soggetto.
- **SVM** = Modello addestrato per **image detection**. Riconosce umani nell'immagine se ne trova le *feature*.

```
import cv2
# Inizializza il descrittore HOG
hog = cv2.HOGDescriptor()

# Imposta l'algoritmo per riconoscere umani
hog.setSVMDetector(
    cv2.HOGDescriptor_getDefaultPeopleDetector()
)

# Cerca umani nell'immagine
(humans, _) = hog.detectMultiScale(
    image,      # I dati dell'immagine
    winStride=(10, 10),
    padding=(24, 24),
    scale=1.05
)
```

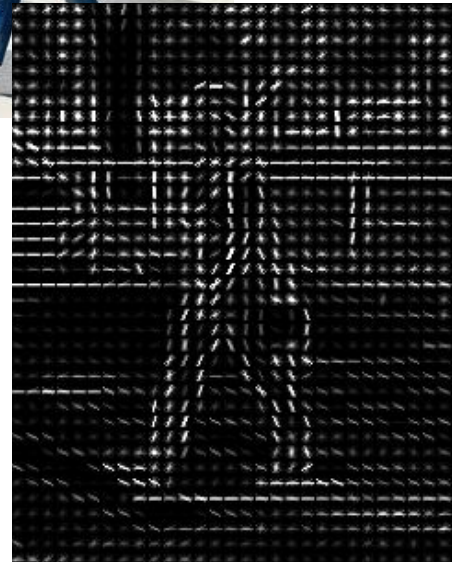
Algoritmo SVM con HOG

HOG: Algoritmo che dà solo informazioni sulla **FORMA** dei soggetti nell'immagine

- Dati legati a gradienti di pixel
- Il PeopleDetector di OpenCV è ottimizzato per immagini con silhouette umane

SVM: prende in input **dati sulla forma** (più semplici da analizzare)

- Analizza i dati e restituisce quanti umani presenti e dove
- Allenato su **training set** di immagini di umani in piedi

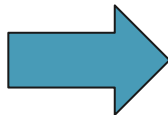


Algoritmo SVM con HOG

Immagine



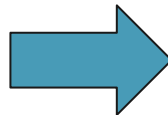
Molti dettagli
Troppi colori



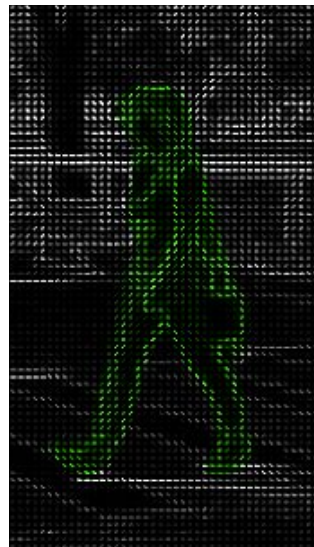
Risultato HOG



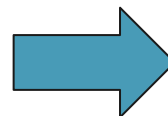
Dati sulla forma del
soggetto
Analisi più semplice



Risultato SVM




ML riconosce
forma "familiare"



Umano
riconosciuto!



E' un umano!



Fasi 3-4: Sperimentazione e analisi

Aprile – Giugno 2023

- Flight Status
- Raccolta dati
- Analisi finale

Fase 3: Flight Status!

Il nostro programma ha passato le selezioni ed è andato sullo spazio!

- 12 maggio 2023
12:46-15:43
- Il sensore a IR ha rilevato dei movimenti
- E la fotocamera ha rilevato...



0 umani



Ma andiamo con ordine...



Fase 4: Analisi dati ricevuti

Un po' di statistiche generali

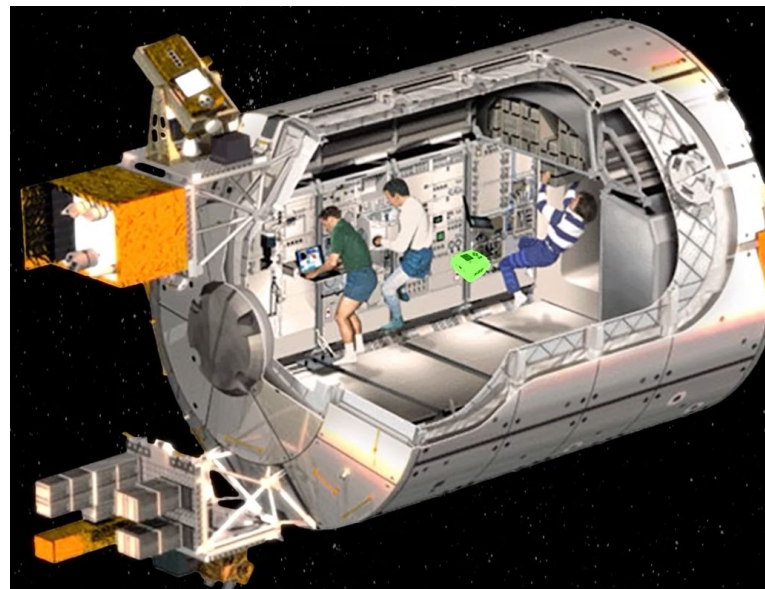
Iterazioni totali	50520
Movimenti rilevati (Sensore IR)	4541
Fotografie scattate	841
● Umani rilevati	0

- **Problema:** circa **98 minuti** passati a scattare foto
- **Conseguenza:** Molti “buchi” nei dati (7 sec. di dati, 10 sec. di pausa)
- **Soluzione per SMA:** Impostare valori del **sensore IR** a 0
- **In futuro:** Usare multithreading per parallelismo

Analisi fotocamera

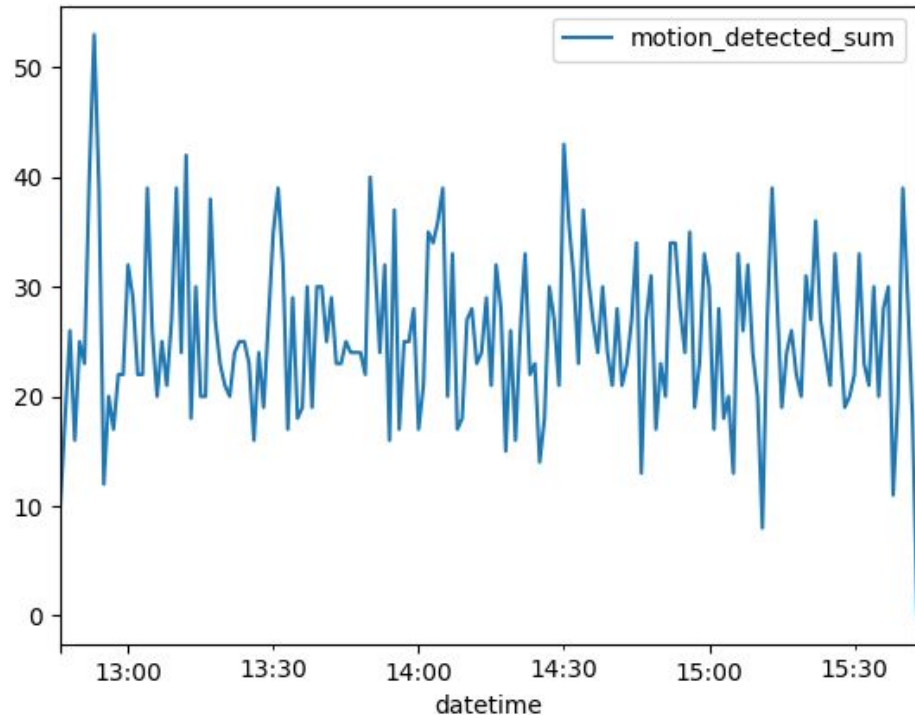
Human detection non ha rilevato umani, ma il sensore sì:

- Fotocamera **non a fuoco** / non inquadrava astronauti
- Astronauti **troppo vicini** a fotocamera: SVM + HOG riconoscono solo silhouette **intere**, **non mezzi busti**
- **Foto sfocate**: astronauti probabilmente in movimento



Analisi su sensore IR (motion_detected)

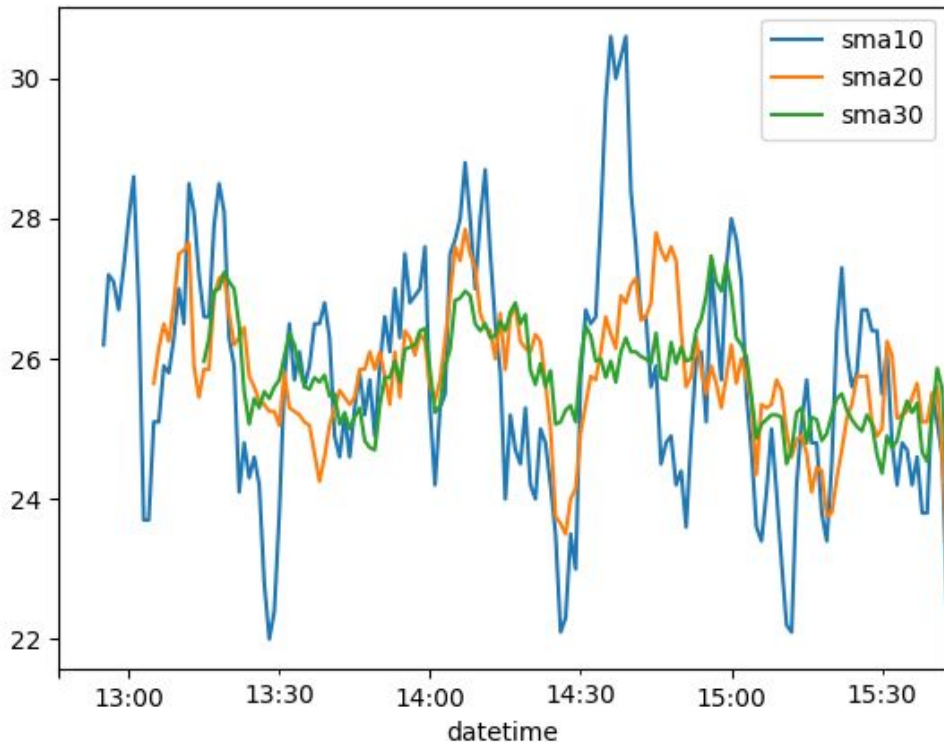
- Movimento rilevato: 10 - 50 volte al min
- Picco prima delle 13:00
- Distribuzione casuale



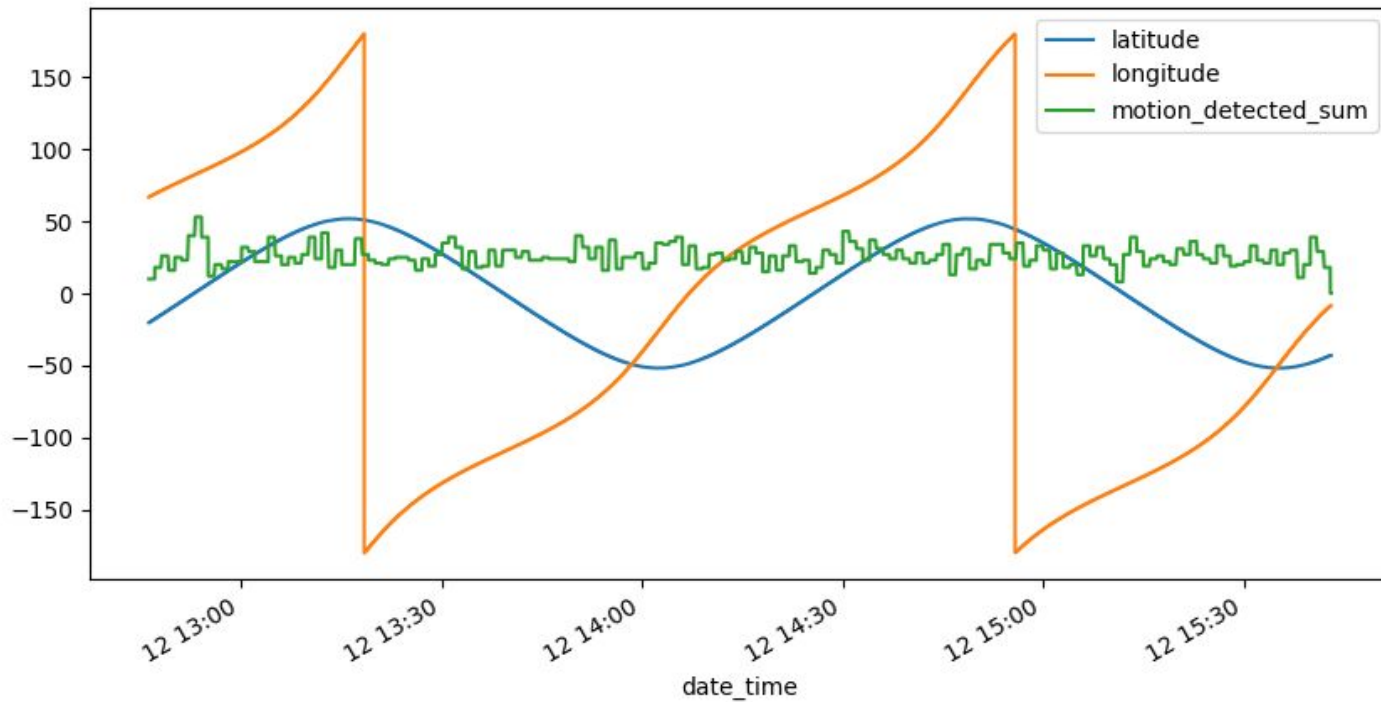
Media mobile su motion_detected

- Media mobile dà risultati?
- Risultati abbastanza casuali con finestre diverse
 - sma10: finestra di 10
 - sma20: finestra di 20
 - sma30: finestra di 30
- Picchi: 13:00, 14:00, 15:00

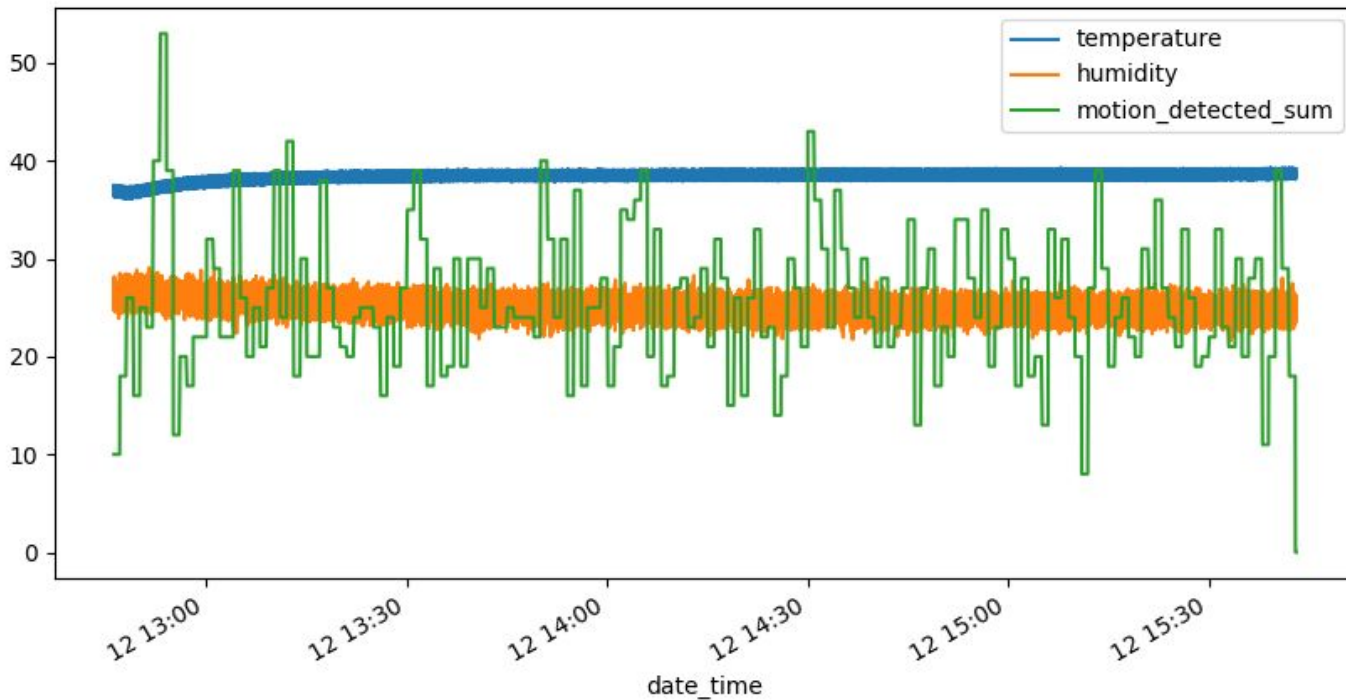
Paragoniamo motion_detected con i valori di altri sensori...



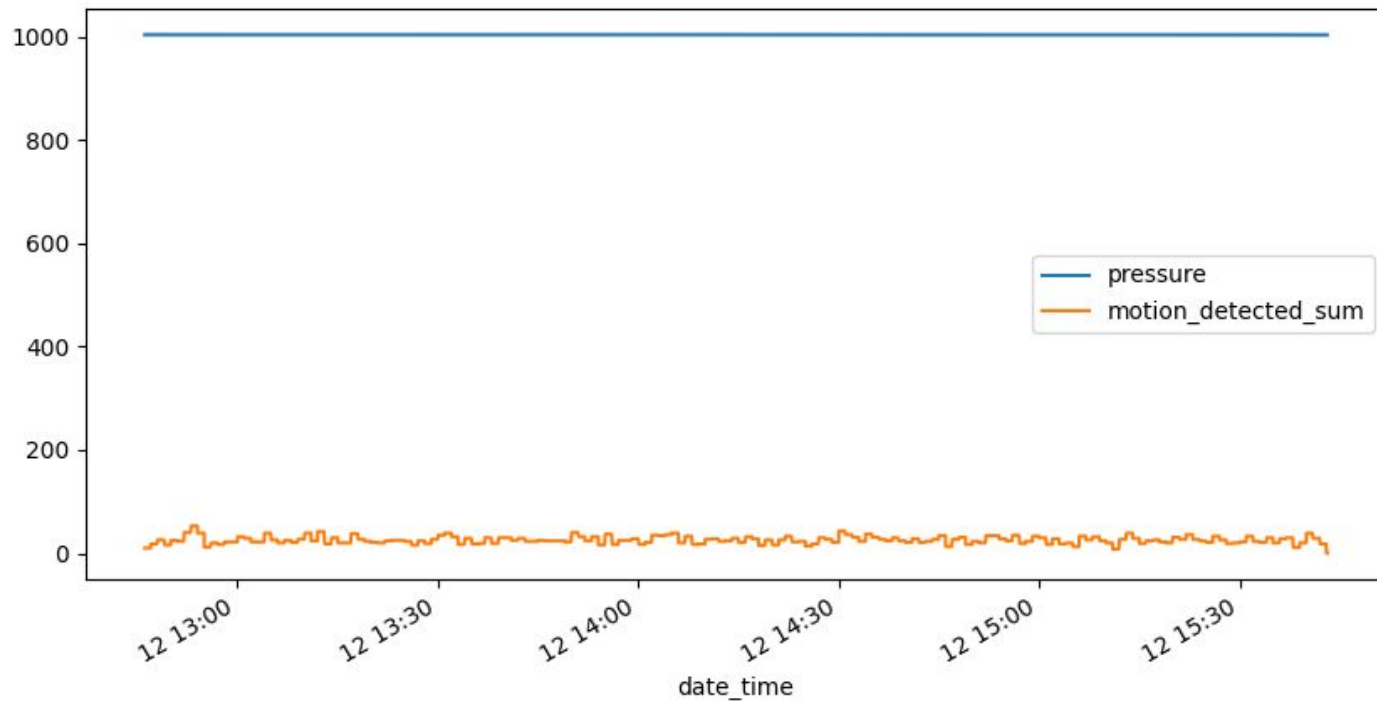
motion_detected e GPS



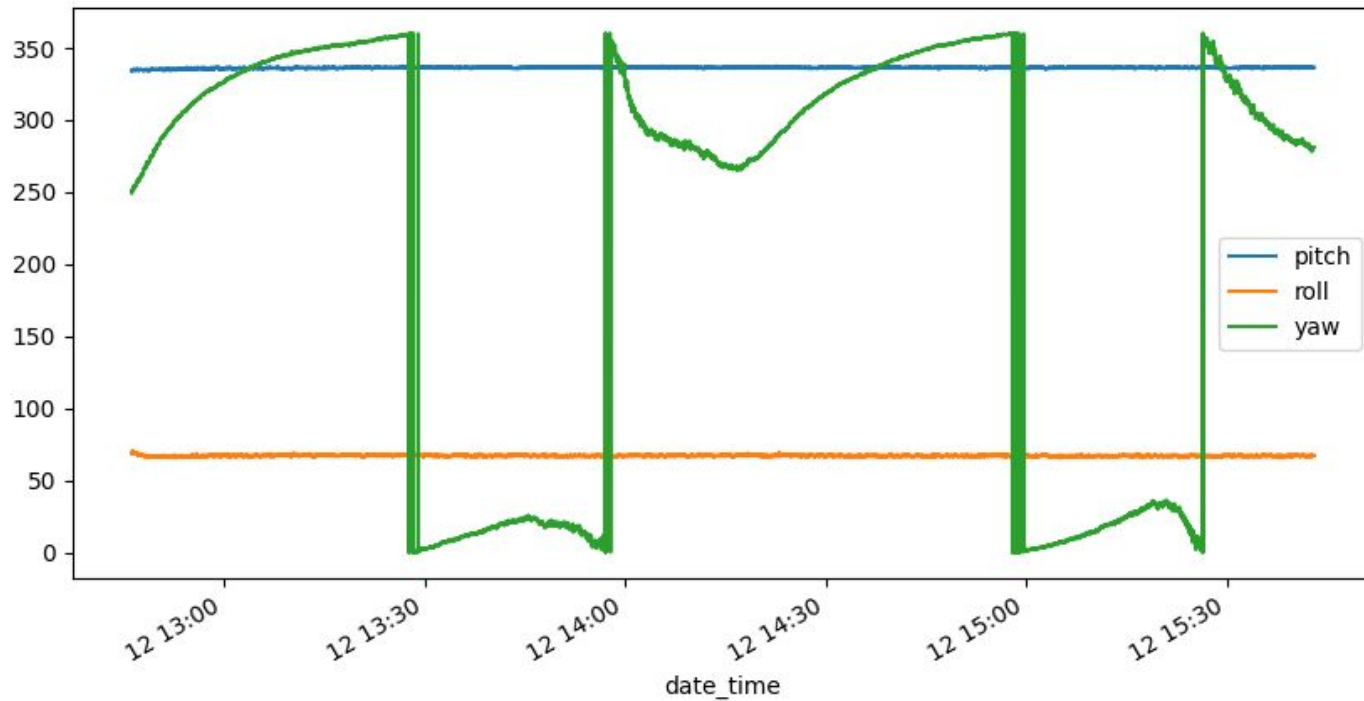
motion_detected e temperatura/umidità



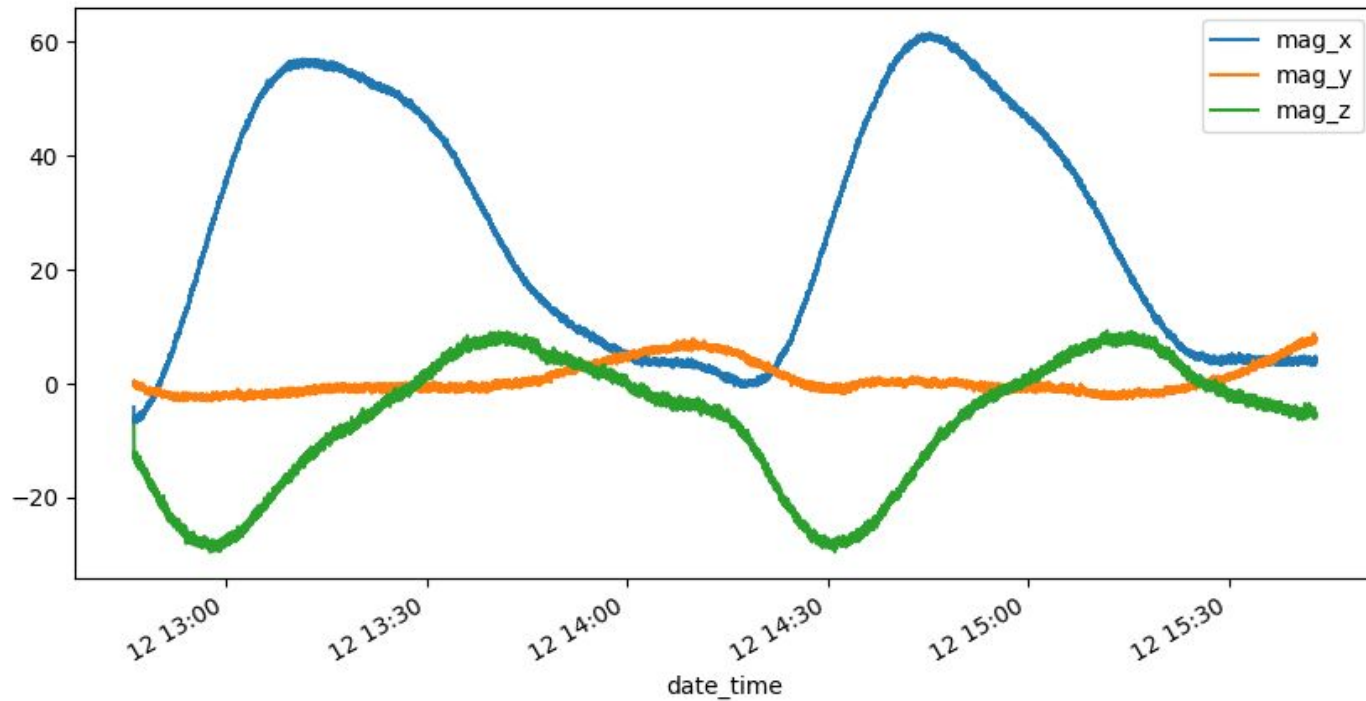
motion_detected e pressione



Orientamento



Magnetometro



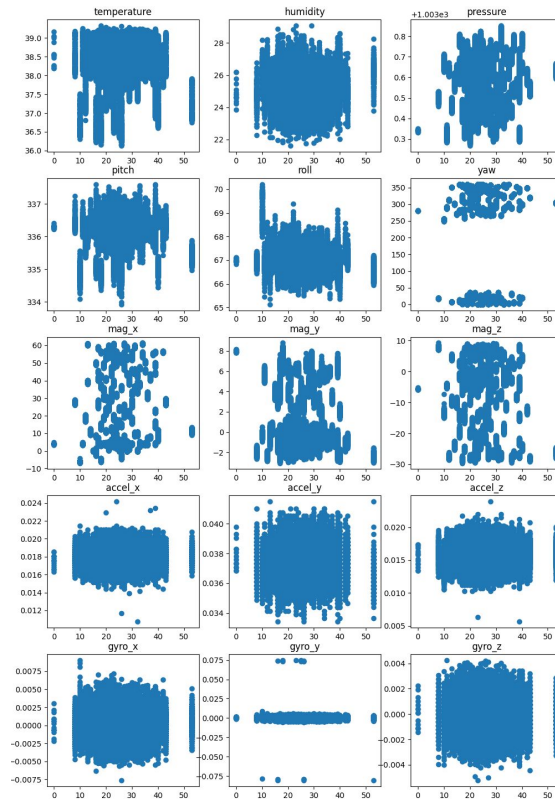
motion_detected vs. altre metriche

Abbiamo calcolato correlazione di Pearson (P) tra motion_detected e altre metriche:

- $-1 < P < 1$
- $0 < |P| < 0.3$: correlazione lineare **debole**
- $0.3 < |P| < 0.7$: correlazione **moderata**
- $0.7 < |P| < 1$: correlazione **forte**

P maggiore trovato: 0.052060

- **Nessuna correlazione** tra motion e altri dati



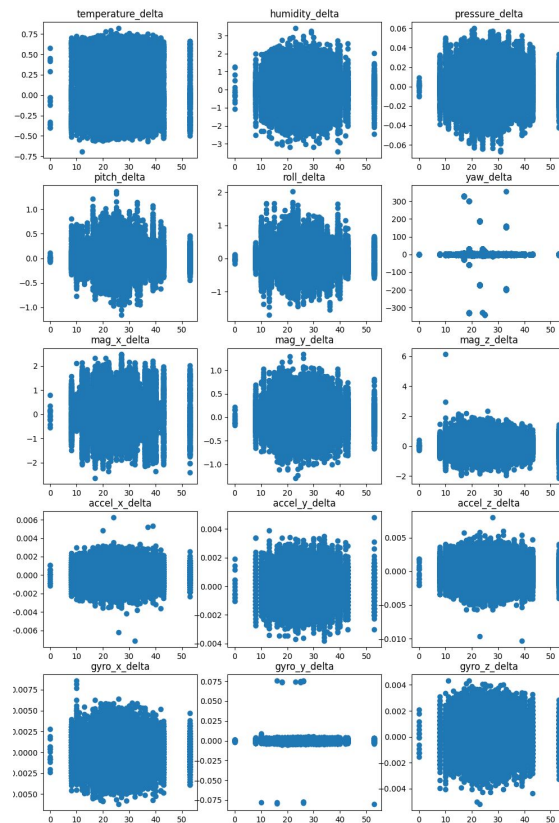
motion_detected vs. delta delle metriche

Correlazione di Pearson tra motion_detected e delta delle metriche

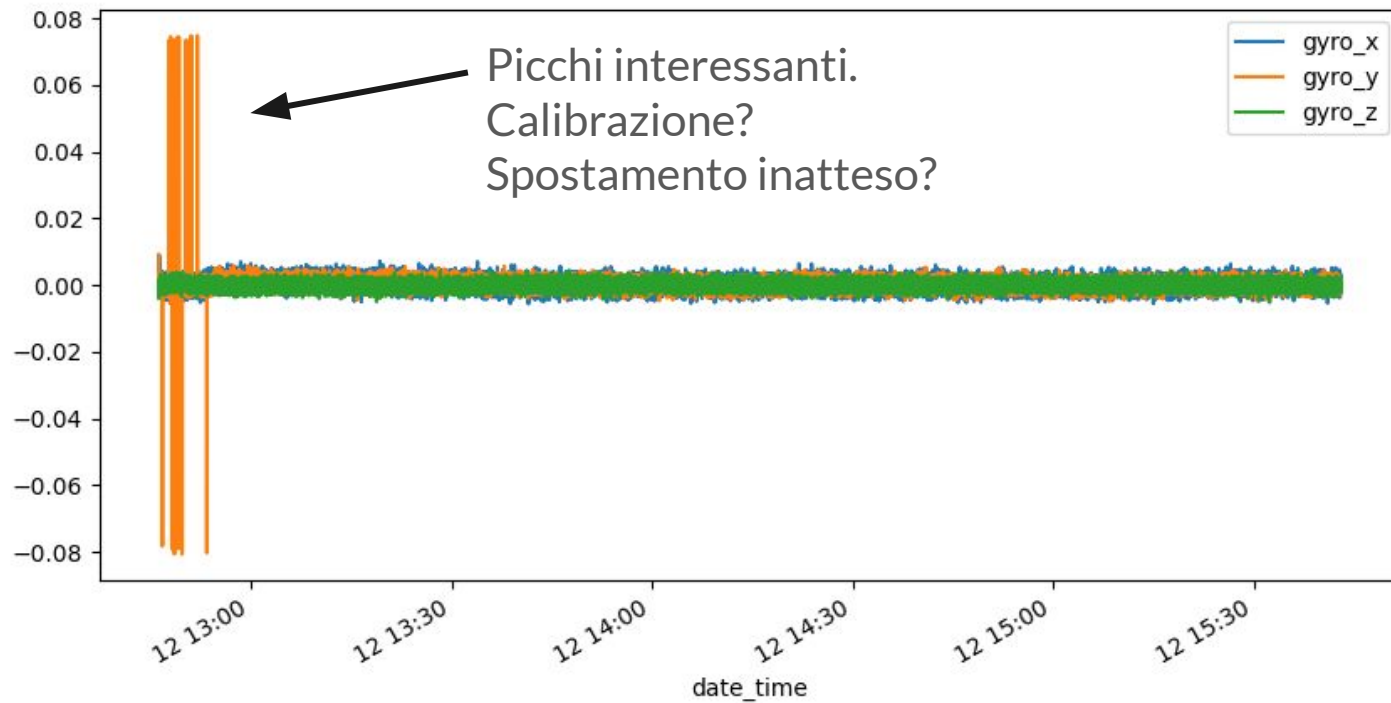
- Si analizza il variare dei dati nel tempo

P maggiore trovato: $8.52 \times 10^{-13} \approx 0$

- **Nessuna correlazione**



Giroscopio



Analisi accelerometro

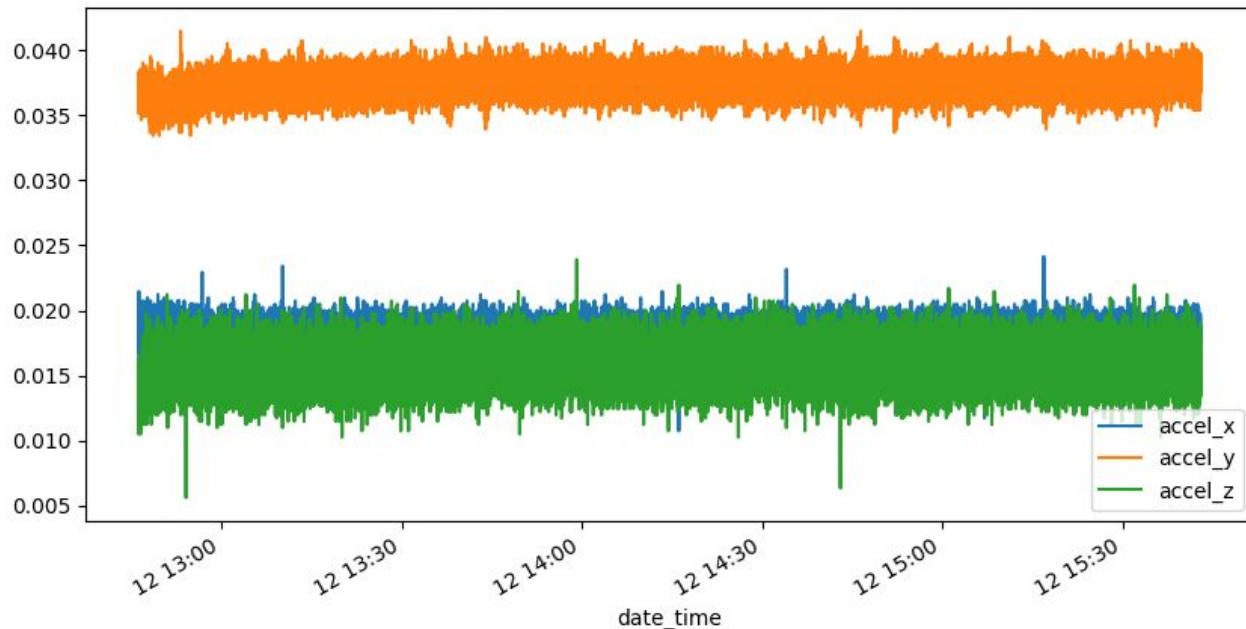
Valori medi:

- x: 0.01795 G
- y: 0.0375 G
- z: 0.01591 G

(Circa 0.1-0.3 m/s²)

Probabili motivi:

- Rumore elettrico
- Turbolenze
- Imprecisione del sensore



Sensibilità accelerometro

Sensore: LSM9DS1

Range: $\pm 2\text{G}$, $\pm 4\text{G}$, $\pm 6\text{G}$, $\pm 8\text{G}$ o $\pm 16\text{G}$

Valori espressi con 16 bit: **65536** val. rappresentabili

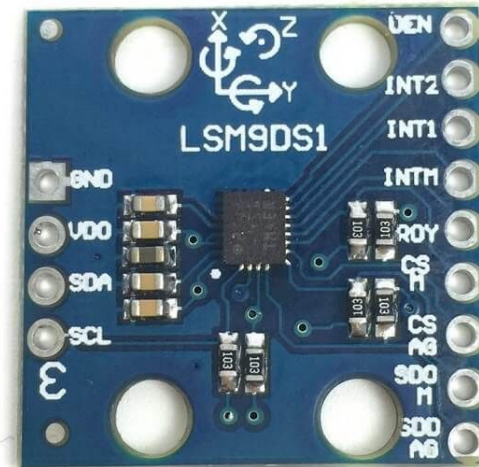
Caso migliore (Range $\pm 2\text{G}$, range di 4000 mG):

- Sensibilità = $4000 \text{ mG} / 65536 = 0.061 \text{ mG}$
= $5.98 \times 10^{-4} \text{ m/s}^2$ (Valore atteso: 10^{-9} m/s^2)

Nel nostro caso (Range $\pm 8\text{G}$): $2.39 \times 10^{-3} \text{ m/s}^2$

→ Sensibilità troppo grande

$$1 \text{ G} = 9.81 \text{ m/s}^2$$



Conclusioni finali

- Rilevata **possibile presenza umana**
- **No conferma** tramite foto o sensori
 - **Accelerometro**: sensibilità troppo grande + rumore
 - **Nessuna correlazione** tra IR e altre metriche
- **Buchi temporali** nei dati

Cosa abbiamo imparato?

- Probabilmente non si può rilevare microgravità
 - Bisogna confermare con dati più esaustivi
- Necessario multithreading per **parallelismo foto / sensori**
- Sensore a ultrasuoni (*distance detection*)?

Links

- Fotografie
 - Foto pedone:
<https://stock.adobe.com/it/images/woman-crossing-street-at-pedestrian-crossing/84780718>
 - Icone: <https://icon-icons.com>
- Repository GitHub:
<https://github.com/pandle/astro-pi-2022-23>



Vi ringrazio per l'ascolto!

@PyDataVenice #15 #Meetup #PyData