Universidade do Minho
Departamento de Informática

SENSORIZAÇÃO E AMBIENTE

MESTRADO EM ENGENHARIA INFORMÁTICA, 1º ANO - Perfil SI

Universidade do Minho
Departamento de Informática

Soft/Physical Sensors

Agenda

- Soft Sensors – PC
  - Keyboard and Mouse
  - PC Usage
- Hands On

Universidade do Minho
Departamento de Informática

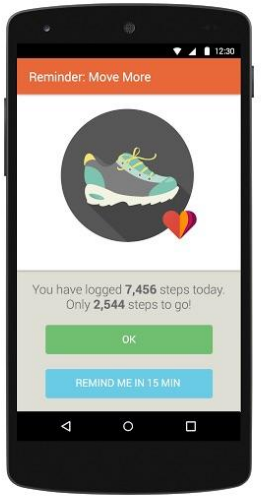
Soft Sensors

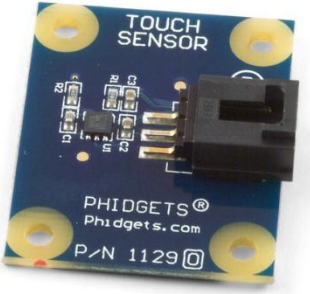
PC

Ambient Intelligence

1. Data Acquisition
    - Sensors
    - Services
    - Data processing
2. Reasoning
    - Data Modeling
    - Machine Learning
    - Decision Models
3. Actuation
    - Notifications
    - Interactions
    - Actions

Sensorization





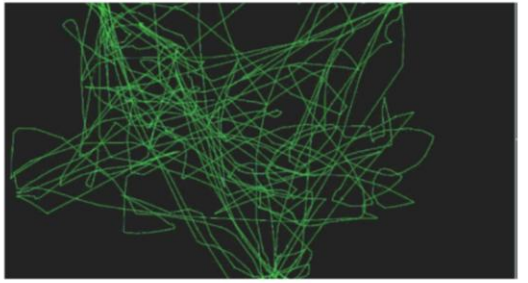
Soft Sensors

Soft Sensors

Keyboard Dynamics KD

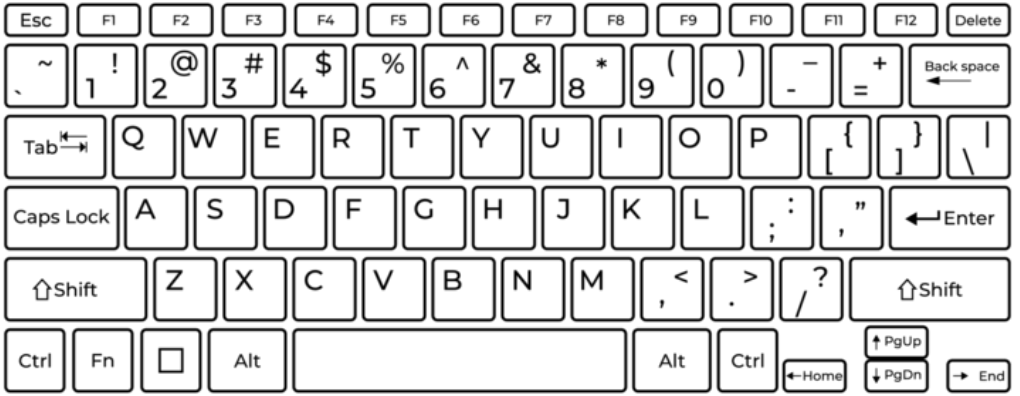
Behavioral Biometrics

MD Mouse Dynamics



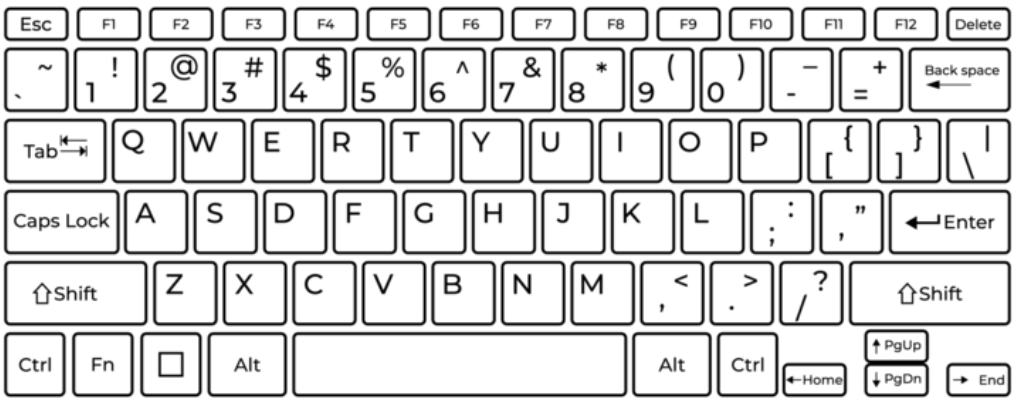
Soft Sensors

- Metrics based on **keystroke events**:



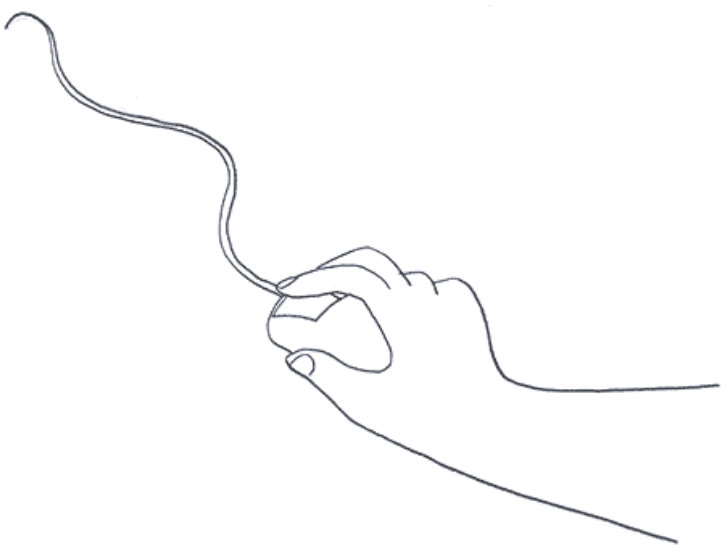
Soft Sensors

- Metrics based on **keystroke events**:
  - Total of pressed characters;
  - Top 5 of pressed characters;
  - Number of characters pressed per minute;
  - Number of times of a character was used;
  - Characters pressed in group (Left Hand, Right Hand and Space);
  - Etc.



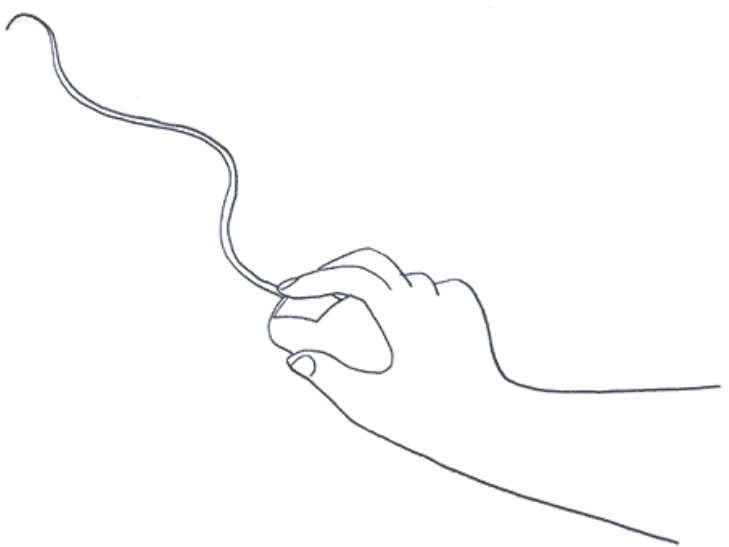
Soft Sensors

- Metrics based on **mouse movement** and **interaction**:

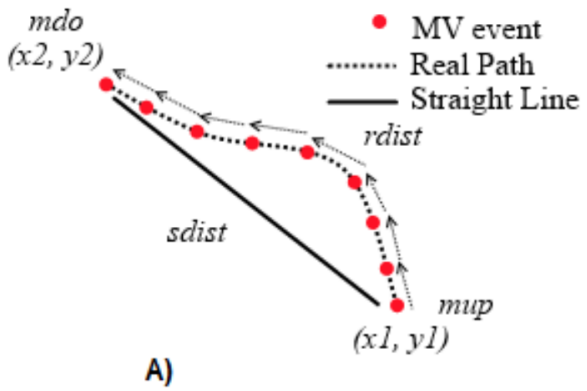


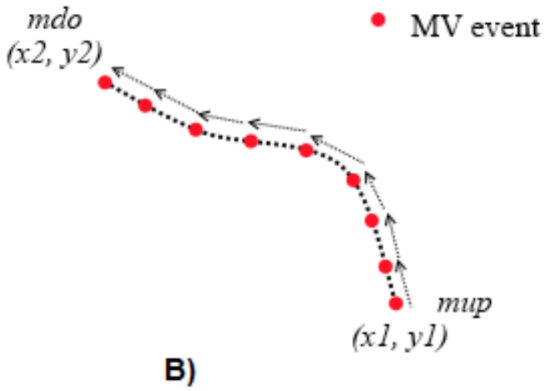
Soft Sensors

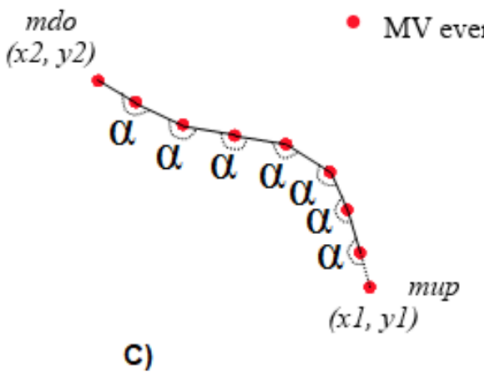
- Metrics based on **mouse movement** and **interaction**:
  - Total distance traveled;
  - Mouse movement speed;
  - Average speed of mouse movements;
  - Number of clicks per second/minute;
  - Etc.

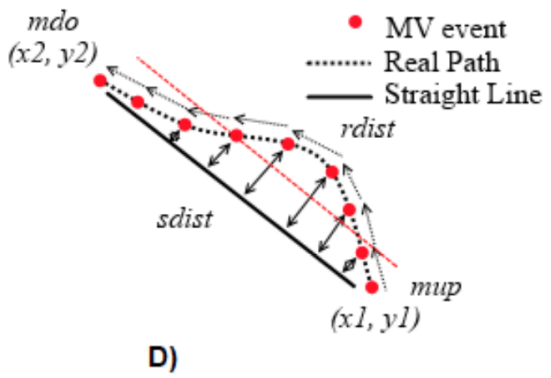


Soft Sensors









Keyboard and Mouse

- Requirements:
  - Python 3.10 or above
  - Framework to run Python code (prompt or IDE such as Anaconda/conda/VS Code/other)
  - Pynput library

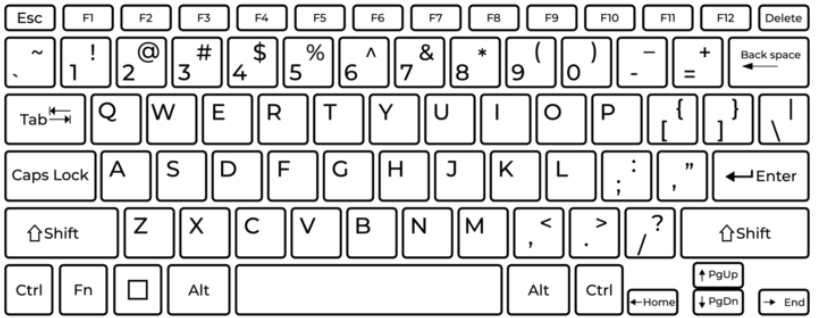
    ```
    pip install pynput
    ```
  - Math library for mouse movement analysis

- Useful links:
  - https://pypi.org/project/pynput/
  - https://pynput.readthedocs.io/en/latest/

Keyboard

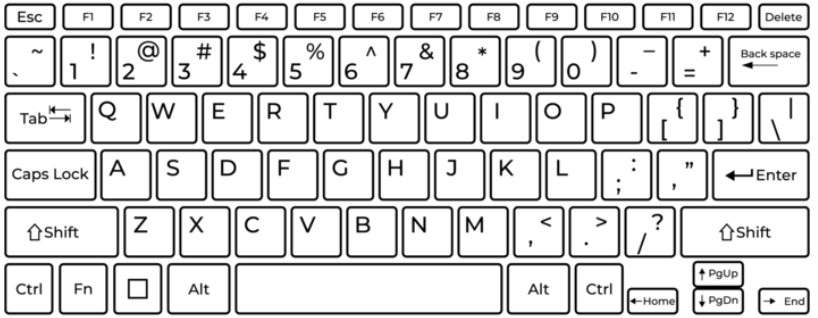
```
from pynput import keyboard                    #for keystroke

#detect key press
def on_press(key):
     try:
          print('alphanumeric key {0} pressed'.format(str(key.char)))
          log_data(datetime.now(),'KeyPressed', str(key.char))
     except:
          print('special key {0} pressed'.format(str(key)))
          log_data(datetime.now(),'SpecialKeyPressed', str(key))
```



Keyboard

```
#detect key releases
def on_release(key):
     print('{0} released'.format(str(key)))
     if key == keyboard.Key.esc:
          try:
               log_data(datetime.now(),'KeyReleased', str(key.char))
          except AttributeError:
               log_data(datetime.now(),'KeyReleased', str(key))
          #stop listener
          print('Gracefully Stopping!')
          return False
```



Keyboard

```
#collecting events
with keyboard.Listener(on_press=on_press, on_release=on_release) as listener:
            listener.join()
#in a non-blocking fashion
listener = keyboard.Listener(on_press=on_press, on_release=on_release)
listener.start()
```

Mouse

```
from pynput import mouse              #for mouse movement

#detect mouse movement
def on_move(x, y):
    print('Pointer moved to {0}'.format((x, y)))
    log_data(datetime.now(),'MouseMovement', str(x) + ',' + str(y))

#detect mouse scroll
def on_scroll(x, y, dx, dy):
    print(‘Mouse scrolled {0} at {1}'.format('down' if dy < 0 else 'up', (x, y)))
    log_data(datetime.now(),'MouseScroll', str(x) + ',' + str(y) + ';' + str(dx) + ',' + str(dy))
```

Mouse

```
#detect mouse click
def on_click(x, y, button, pressed):
    print('{0} at {1}'.format('Pressed' if pressed else 'Released', (x, y)))
    log_data(datetime.now(),'MouseClicked', str(button))
    if not pressed:
        #stop listener
        print('Gracefully Stopping!')
        return False

#collecting events
with mouse.Listener(on_move=on_move, on_click=on_click, on_scroll=on_scroll) as listener:
       listener.join()
#in a non-blocking fashion
listener = mouse.Listener(on_move=on_move, on_click=on_click, on_scroll=on_scroll)
listener.start()
```

Logging

- For each detection (keystroke or mouse movement):
  - Open a file, with the append flag:

```
f = open('logger.txt', 'a')
```

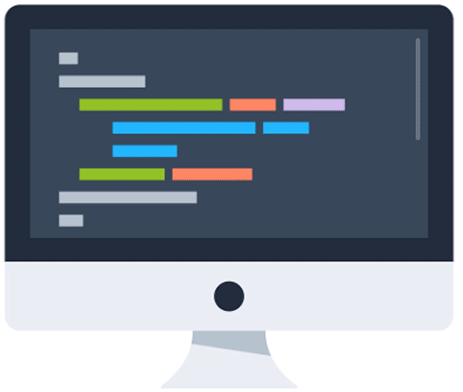
  - Write the detected event to the file:

```
f.write(str(timestamp) + '|' + event + '|' + value + '\n')
```

  - Close the file:

```
f.close()
```

- Import `datetime` and `math`



Logging

```
[TIMESTAMP]|[EVENT]|[VALUE]
2025-02-19 10:42:51|KeyPressed|d
2025-02-19 10:42:51|KeyPressed|k
2025-02-19 10:42:52|KeyPressed|b
2025-02-19 10:42:52|KeyPressed|g
2025-02-19 10:42:53|MouseMovement|794,-439
2025-02-19 10:42:53|MouseMovement|796,-439
2025-02-19 10:42:53|MouseMovement|799,-438
2025-02-19 10:42:54|MouseClicked|left
…
```

PC Usage

- But more can be done. For example, what about network/cpu/memory usage?

- Requirements:
  - Psutil library:

```
pip install psutil
```

- `psutil` *(process and system utilities)* is a cross-platform library for retrieving information on running processes and system utilization (CPU, memory, disks, network, sensors) in Python
  - Useful mainly for system monitoring and profiling
  - Implements functionalities offered by classic UNIX command line tools such as `ps`, `top`, `iotop`, `lsof`, `netstat`, `ifconfig`, `free` and others

https://pypi.org/project/psutil/

PC Usage

```
import psutil
'''
CPU
'''
print(f"CPU Times: {psutil.cpu_times()}")
print(f"CPU Count: {psutil.cpu_count()}")
print(f"Physical CPU Count: {psutil.cpu_count(logical=False)}")
print(f"CPU Stats: {psutil.cpu_stats()}")
print(f"CPU Frequency: {psutil.cpu_freq()}")

for x in range(3):
    print(f"CPU Usage: {psutil.cpu_percent(interval=1)}")
```

PC Usage

```
import psutil
'''
MEMORY
'''
print(f"Virtual Memory: {psutil.virtual_memory()}")
print(f"Swap Memory: {psutil.swap_memory()}")

'''
DISKS
''’
print(f"Partitions: {psutil.disk_partitions()}")
print(f"Disk usage (from /): {psutil.disk_usage('/')}")
```

PC Usage

```
import psutil
'''
NETWORK
'''
print(f"Addresses: {psutil.net_if_addrs()}")
print(f"Net stats: {psutil.net_if_stats()}")

'''
OTHER SENSORS
'''
try:
    print(f"Temperature: {psutil.sensors_temperatures()}")
    print(f"Fans: {psutil.sensors_fans()}")
except:
    print(f"Linux only!")

print(f"Battery: {psutil.sensors_battery()}")
```

Universidade do Minho
Departamento de Informática

Hands On

Hands On

- **For even student numbers**
  1. Collect **keyboard** data, create a log file with the collected data and calculate relevant metrics;
  2. Send the raw data to a **feed** using Firebase.

- **For odd student numbers**
  1. Collect **mouse** data, create a log file with the collected data and calculate relevant metrics;
  2. Send the raw data to a **feed** using Firebase.

https://firebase.google.com/docs/firestore/quickstart#python