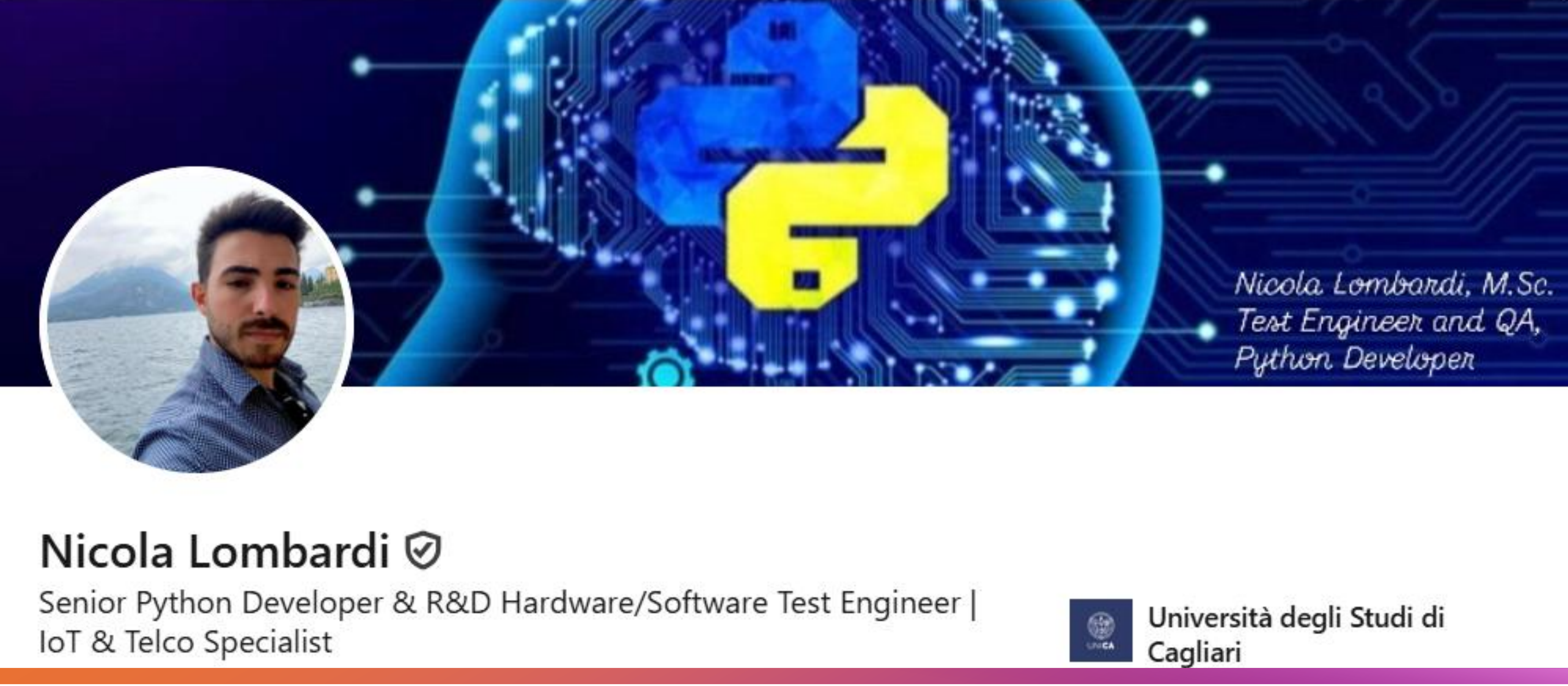


Nicola Lombardi, M.Sc.
Test Engineer and QA,
Python Developer

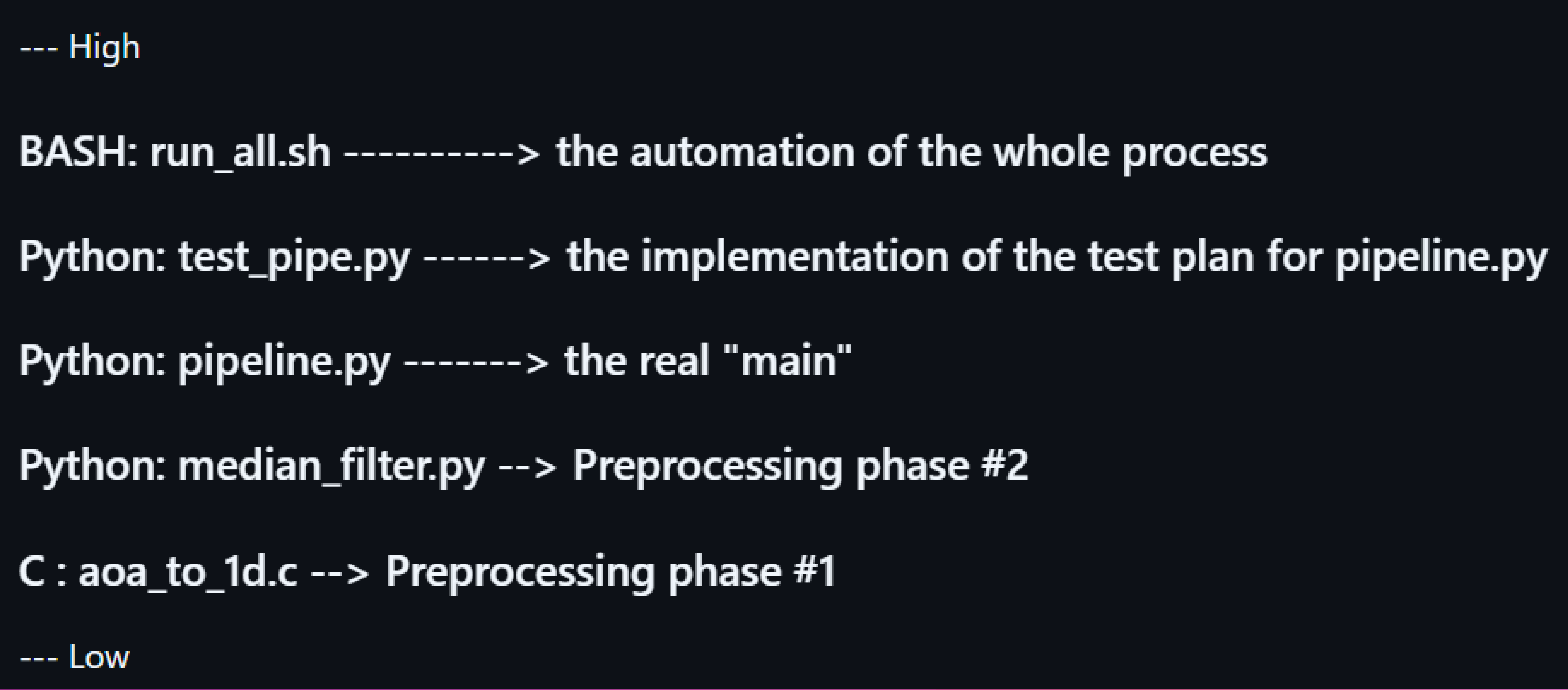
Nicola Lombardi

Senior Python Developer & R&D Hardware/Software Test Engineer | IoT & Telco Specialist

Università degli Studi di Cagliari

A real time system with Pipeline

Tracking system using C and Python



--- High

BASH: run_all.sh ----------> the automation of the whole process

Python: test_pipe.py ------> the implementation of the test plan for pipeline.py

Python: pipeline.py -------> the real "main"

Python: median_filter.py --> Preprocessing phase #2

C : aoa_to_1d.c --> Preprocessing phase #1

--- Low

The system design

Nicola Lombardi, Senior Test Engineer

Why Python and C language?

It is necessary to have a very fast system: therefore text files cause latency, even C is much faster than Python and Java.

We need the so-called "Pipes" in a concurrency environment where a lot of data arrives to be stored.

The record is given by " &timestamp, tag_id, &angle, &h_tag " and each record represents a moving cow.

The system must detect if there are changes in movement and update them, but if the system has collisions or invalid data, it must detect them immediately.

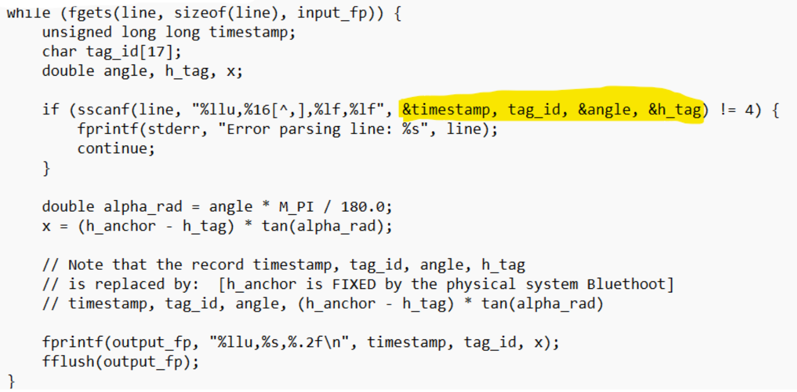
If there are anomalous positions, how do we detect False Positives? We use an algorithm through a sliding time window.

Here Python comes into play, which is very powerful in terms of calculation, but slow.

So why use it? Its calculation power is unmatched, so we coordinate C (the Data Record Register that takes care of storing data inside the pipes) with Python (Processing Unit).

Nicola Lombardi, Senior Test Engineer

The algorithm explained in simple way



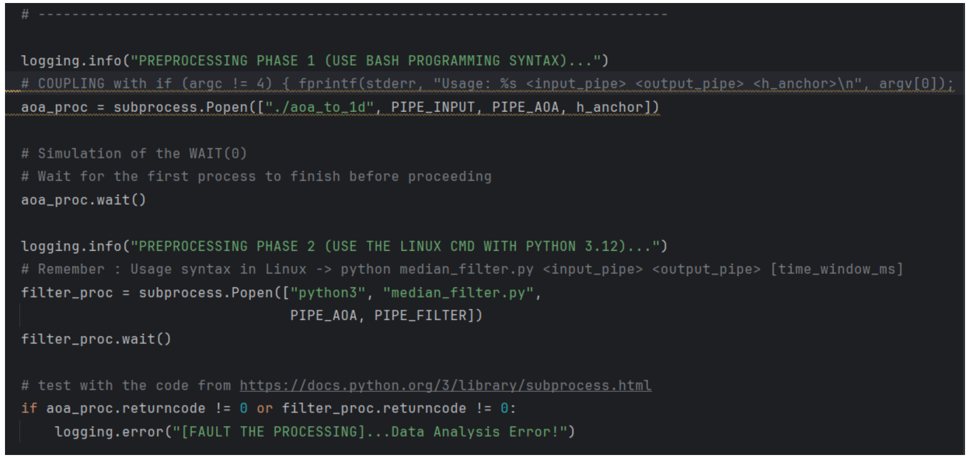
```c
while (fgets(line, sizeof(line), input_fp)) {
    unsigned long long timestamp;
    char tag_id[17];
    double angle, h_tag, x;

    if (sscanf(line, "%llu,%16[^,],%lf,%lf", &timestamp, tag_id, &angle, &h_tag) != 4) {
        fprintf(stderr, "Error parsing line: %s", line);
        continue;
    }

    double alpha_rad = angle * M_PI / 180.0;
    x = (h_anchor - h_tag) * tan(alpha_rad);

    // Note that the record timestamp, tag_id, angle, h_tag
    // is replaced by:  [h_anchor is FIXED by the physical system Bluethoot]
    // timestamp, tag_id, angle, (h_anchor - h_tag) * tan(alpha_rad)

    fprintf(output_fp, "%llu,%s,%.2f\n", timestamp, tag_id, x);
    fflush(output_fp);
}
```

Process #1:

Aoa_to_1d.c

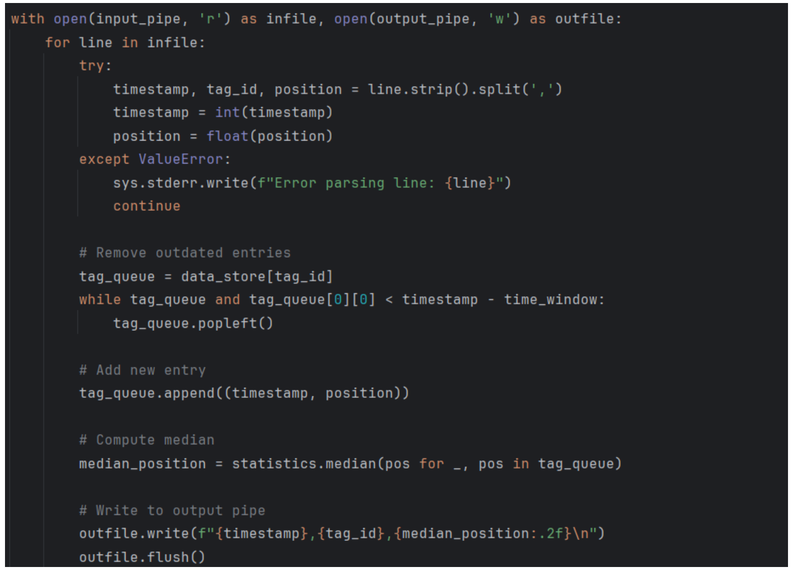
P1

Scheduler [Python]



```python
# ---------------------------------------------------------------------------

logging.info("PREPROCESSING PHASE 1 (USE BASH PROGRAMMING SYNTAX)...")
# COUPLING with if (argc != 4) { fprintf(stderr, "Usage: %s <input_pipe> <output_pipe> <h_anchor>\n", argv[0]);
aoa_proc = subprocess.Popen(["./aoa_to_1d", PIPE_INPUT, PIPE_AOA, h_anchor])

# Simulation of the WAIT(0)
# Wait for the first process to finish before proceeding
aoa_proc.wait()

logging.info("PREPROCESSING PHASE 2 (USE THE LINUX CMD WITH PYTHON 3.12)...")
# Remember : Usage syntax in Linux -> python median_filter.py <input_pipe> <output_pipe> [time_window_ms]
filter_proc = subprocess.Popen(["python3", "median_filter.py",
                                PIPE_AOA, PIPE_FILTER])
filter_proc.wait()

# test with the code from https://docs.python.org/3/library/subprocess.html
if aoa_proc.returncode != 0 or filter_proc.returncode != 0:
    logging.error("[FAULT THE PROCESSING]...Data Analysis Error!")
```

Dispatcher



```python
with open(input_pipe, 'r') as infile, open(output_pipe, 'w') as outfile:
    for line in infile:
        try:
            timestamp, tag_id, position = line.strip().split(',')
            timestamp = int(timestamp)
            position = float(position)
        except ValueError:
            sys.stderr.write(f"Error parsing line: {line}")
            continue

        # Remove outdated entries
        tag_queue = data_store[tag_id]
        while tag_queue and tag_queue[0][0] < timestamp - time_window:
            tag_queue.popleft()

        # Add new entry
        tag_queue.append((timestamp, position))

        # Compute median
        median_position = statistics.median(pos for _, pos in tag_queue)

        # Write to output pipe
        outfile.write(f"{timestamp},{tag_id},{median_position:.2f}\n")
        outfile.flush()
```

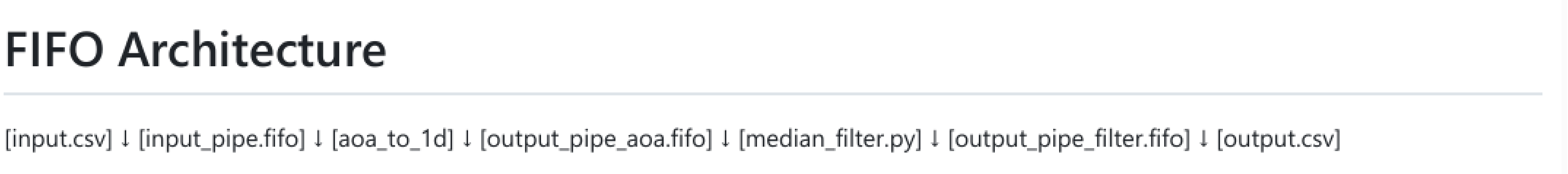
Process #2:

Median_filter.py

P2

Nicola Lombardi, Senior Test Engineer

Data storaging arrangement



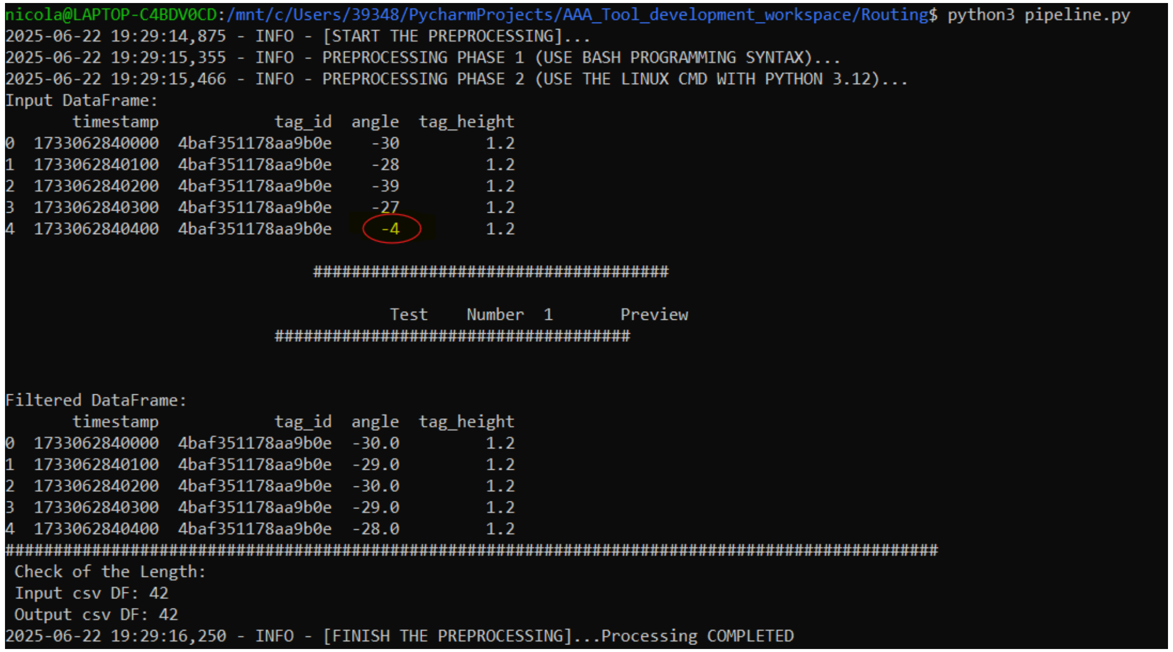
FIFO Architecture

[input.csv] ↓ [input_pipe.fifo] ↓ [aoa_to_1d] ↓ [output_pipe_aoa.fifo] ↓ [median_filter.py] ↓ [output_pipe_filter.fifo] ↓ [output.csv]

Nicola Lombardi, Senior Test Engineer

Nicola Lombardi, Senior Test Engineer

Demo



```
nicola@LAPTOP-C4BDV0CD:/mnt/c/Users/39348/PycharmProjects/AAA_Tool_development_workspace/Routing$ python3 pipeline.py
2025-06-22 19:29:14,875 - INFO - [START THE PREPROCESSING]...
2025-06-22 19:29:15,355 - INFO - PREPROCESSING PHASE 1 (USE BASH PROGRAMMING SYNTAX)...
2025-06-22 19:29:15,466 - INFO - PREPROCESSING PHASE 2 (USE THE LINUX CMD WITH PYTHON 3.12)...
Input DataFrame:
       timestamp           tag_id  angle  tag_height
0  1733062840000  4baf351178aa9b0e    -30         1.2
1  1733062840100  4baf351178aa9b0e    -28         1.2
2  1733062840200  4baf351178aa9b0e    -39         1.2
3  1733062840300  4baf351178aa9b0e    -27         1.2
4  1733062840400  4baf351178aa9b0e     -4         1.2

                                #####################################

                                        Test    Number  1       Preview
                            #####################################


Filtered DataFrame:
       timestamp           tag_id  angle  tag_height
0  1733062840000  4baf351178aa9b0e  -30.0         1.2
1  1733062840100  4baf351178aa9b0e  -29.0         1.2
2  1733062840200  4baf351178aa9b0e  -30.0         1.2
3  1733062840300  4baf351178aa9b0e  -29.0         1.2
4  1733062840400  4baf351178aa9b0e  -28.0         1.2
#################################################################################################
 Check of the Length:
 Input csv DF: 42
 Output csv DF: 42
2025-06-22 19:29:16,250 - INFO - [FINISH THE PREPROCESSING]...Processing COMPLETED
```
)