NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DTSD Lab06

dloTspmatrix: distributed Sparse Matrix processing on resource-constrained IoT devices

v1.0

Pedro Maló

pmm@fct.unl.pt

2st stage: FULL dIoTspmatrix

(for evaluation)

dIoTspmatrix digital system

dIoTspmatrix digital system: distributed Sparse Matrix processing on resource-constrained IoT devices

- Distributed processing: distributed processing of operations sparse matrices among several processing nodes
- Matrix and Sparse Matrix operations: compute typicall operations of Matrices (e.g., transpose, add, multiply, etc.) and Sparse Matrices (e.g., eye, compress, etc.)
- Resource-constrained devices: execute Sparce Matrices operations on resource-constrained devices (limited memory, limited performance, limited power)
- Internet-of-Things: make use IoT-based communications (e.g. MQTT)

dIoTspmatrix: Development approach

1st stage: BASIC dIoTspmatrix (not for evaluation)

- Implementation using DOK (Dictionary Of Keys) and basic set sparse matrix operations
- Programming Goal: **Basic Python**
  - Individual assignment (by each student)
  - Procedural Programming with Python
  - Basic software testing using pytest
- Programming Approach: **Code-first, test-last**
  - A series of public tests will be made available to students
  - A series of private tests are executed periodically and the results are made available to the the students (private tests source code is not made available)

2nd stage: FULL dIoTspmatrix (evaluation)

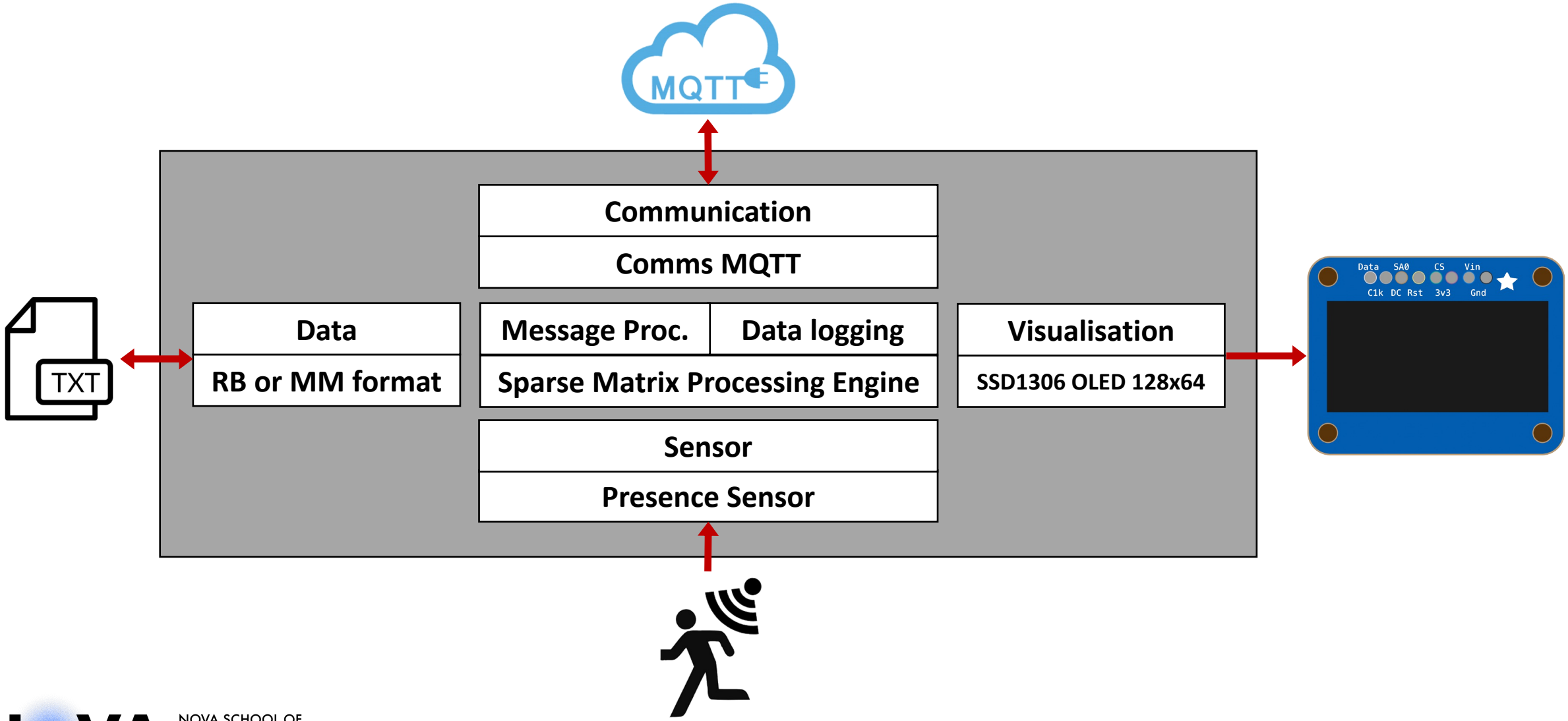
- Implementation using additional sparse matrix representations and a wide set of operations
- Programming Goal: **Advanced Python**
  - Group assignment (2~3 students per group)
  - Object-Oriented Programming Python
  - Advanced testing with pytest & unittest
- Program approach: **Test-first, code-after**
  - Initial set of public tests provided for initial software development
  - New public tests made available periodically to students for software code refactoring
  - Batch of exhaustive private tests will be executed at the end of the assignment

dIoTspmatrix Scenario

Monitor the number of people entering/leaving an office meeting room

- Log the number of people using a **motion sensor** (PIR motion sensor)
- **Record logs each minute in a matrix** with 24 lines [0:23] representing hours and 60 columns [0:59] representing minutes.
  - Each matrix element encloses the number of occurrences detected by the motion sensor.
- **Representation of the matrix is sparse** as people in/out the meeting room only from time to time and also the meeting room is not used in off-office hours
- Communicate logs with others with **MQTT** using **“normalised” data representation** for sparse matrices (compressed format)
- Display data (locally, at the edge) in a visualisation dashboard (OLED) using a pixelized **heatmap** representation.

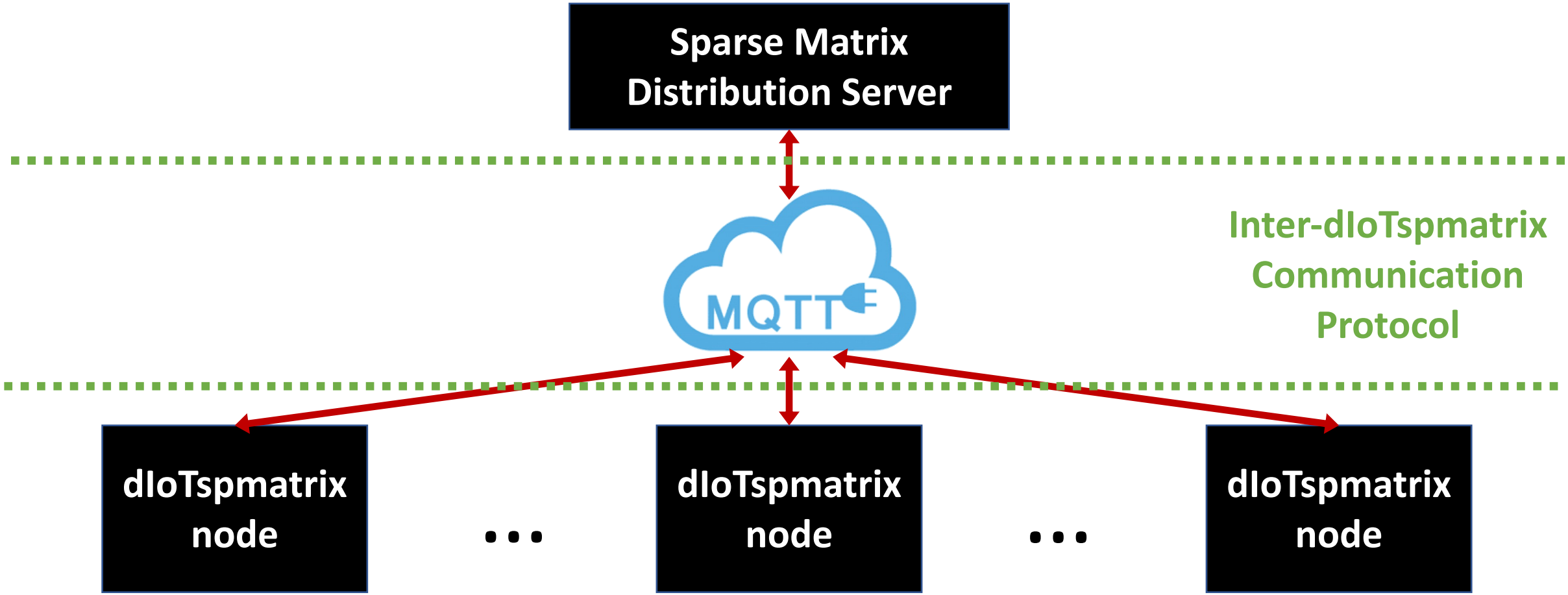
dIoTspmatrix Node: Logical Architecture



Presence log (Sparse Matrix)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	42	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59
0																																																															
1																																																																					
2																																																																					
3																																																																					
4																																																																					
5																																																																					
6																																																																					
7																																																																					
8																																																																					
9																																																																					
10																																																																					
11																																																																					
12																																																																					
13																																																																					
14																																																																					
15																																																																					
16																																																																					
17																																																																					
18																																																																					
19																																																																					
20																																																																					
21																																																																					
22																																																																					
23																																																																					

dIoTspmatrix Communication Protocol



Communication Protocol JSON Messages

Request: day data log

{ **“msg”**: “get”

“id”: “<device id>”

}

Reply: day data log

{ **“id”**: ”<device id>”

“data”: “<day data log as sparse matrix compressed representation>”

}

Request: sum of data logs

{ **“msg”**: “add”

“id”: ”<device id>”

“m1”: “<spmatrix in compressed representation>”

“m2”: “<spmatrix compressed representation>”

}

Reply: sum of data logs

{ **“msg”**: “add”

“id”: ”<device id>”

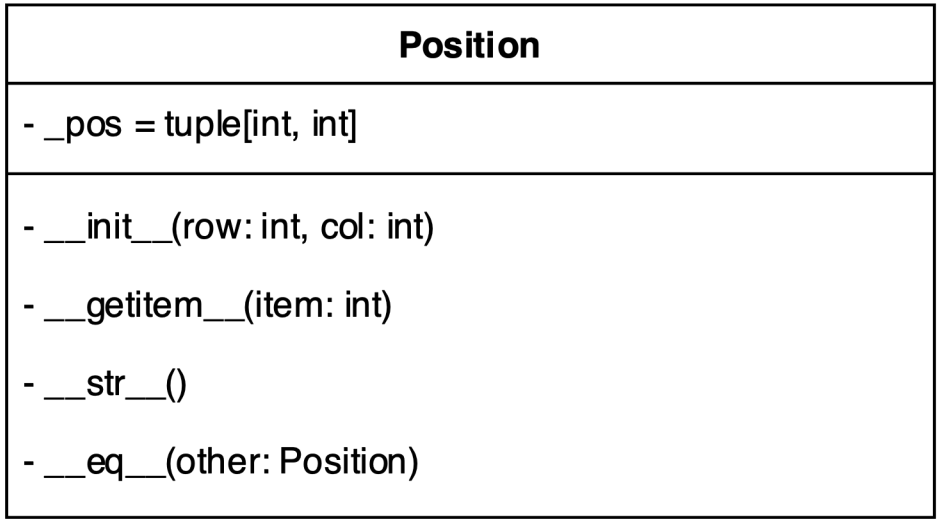
“add”: “<spmatrix in compressed representation>”

}

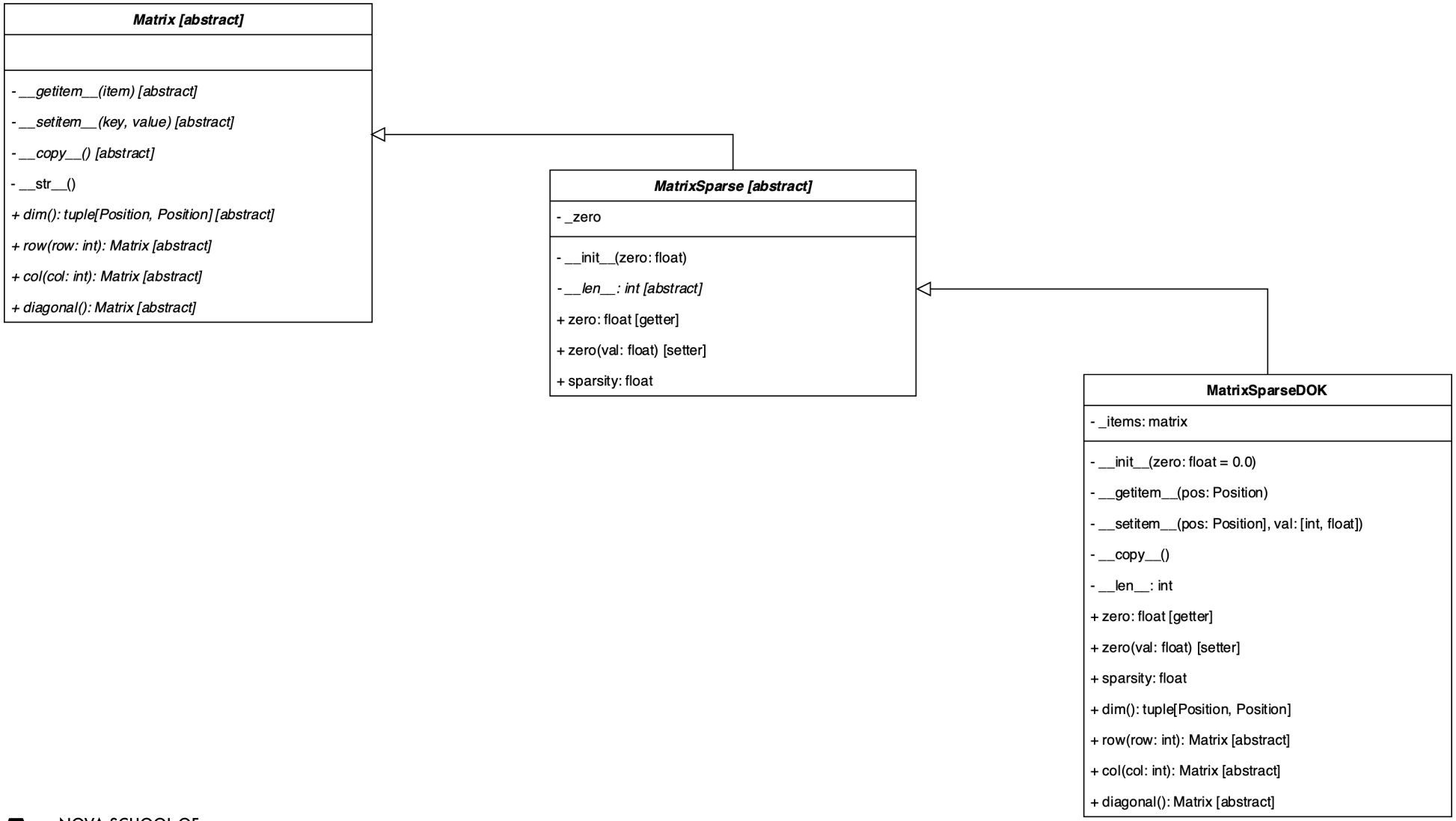
Procedural → Object Oriented

- '*_create' methods → class __init__ special method
- '*_is' methods → no longer needed, realised by object context
- 'position_row' & 'position_col' → __getitem__ special methods
- '*_value_get' & '*_value_set' → __getitem__ & __setitem__ special methods
- '*_zero_get' & '*_zero_set' methods → python getters & setters
- '*_copy' → __copy__ special method
- '*_str' → __str__ special method
- '*_equal' → __eq__ special method

Sparse Matrix: Procedural → Object Oriented



Position: Procedural → Object Oriented



Specification: New Features (1/2)

- **Iterate over the elements of the sparse matrix:**
  - Iterate only over the non-null/-zero elements of the sparse matrix
  - Iterate positions ordered by row then by column
  - Operation is implemented via the python __iter__ & __next__ special methods
- **Compare one sparse matrix to another:**
  - Operation is implemented via the python __eq__ special method
- **Add to sparse matrix**:
  - Add a number to a sparse matrix or add two sparse matrices
  - Sparse matrices can only be added if are of the same dimension and have the same zero/null
    - If arguments invalid => raise exception ValueError with message '_add_matrix() incompatible matrices'
  - Operation is implemented via the python __add_ special method
- **Multiply to sparse matrix**:
  - Multiply a number to a sparse matrix or multiply two sparse matrices
  - Sparse matrices can only be multiplied if have compatible dimensions and have the same zero/null
    - If arguments invalid => raise exception ValueError with message '_mul_matrix() incompatible matrices'
  - Operation is implemented via the python __mul_ special method
- **def transpose(self) -> MatrixSparse**
  - Method to transpose a Sparse Matrix

Specification: New Features (2/2)

- **def eye(size: int, unitary: float = 1.0, zero: float = 0.0) -> SparseMatrix**
  - Static method to create a square identity sparse matrix
  - Return a size-by-size identity matrix with unitary values on the main diagonal and zero values elsewhere
  - By default, unitary value is 1.0 and zero/null value is 0.0
- **def compress() -> compressed (i.e., tuple[[int, int], float, tuple[float], tuple[int], tuple[int]])**
  - Method to compresses a Sparse Matrix with the double-offset indexing algorithm (see ahead)
  - Returns a tuple of 5 elements (upper left corner position, zero/null value, value vector, index vector, offset vector)
  - If matrix is empty => raise exception ValueError with message ‘compress() empty sparse matrix’
- **def doi(comp_vector: compressed, pos: Position) -> float**
  - Static method to get a value out of a sparse matrix using its compressed representation
  - comp_vector is the 5-elements tuples sparse matrix compressed representation (out of the compress)
    - If comp_vector is not a compressed ADT => raise exception ValueError with message ‘doi() invalid parameters’
  - pos is the Position whose value in the sparse matrix is to be returned
- **def decompress(comp_vector: compressed) -> MatrixSparse**
  - Fully decompresses a sparse matrix out of its compressed representation
  - comp_vector is the 5-elements tuples sparse matrix compressed representation (out of the compress)
    - If comp_vector is not a compressed ADT => raise ValueError with message ‘decompress() invalid parameters’

Evaluation Scoring

- Sparse Matrix Object Oriented Private Tests **(10 points)**
  - *Especially focusing on the compress, doi, decompress methods*

- Implementation of the Presence Sensor Interface **(2 point)**
  - Including implementation of data logging capability

- Implementation of the Communication Protocol **(5 points)**
  - Able to send & receive (and process) dIoTspmatrix messages

- Implementation of Data import/export **(2 points)**
  - RB or MM file format (interfaced via SD Card in wokwi)

- Implementation of Visualisation Dashboard **(1 points)**
  - Dashboard with heatmap of on SSD1306 OLED 128x64