NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DTSD Lab06

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

v1.4

Pedro Maló

pmm@fct.unl.pt

2st stage: FULL dIoTspmatrix

(for evaluation)

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)

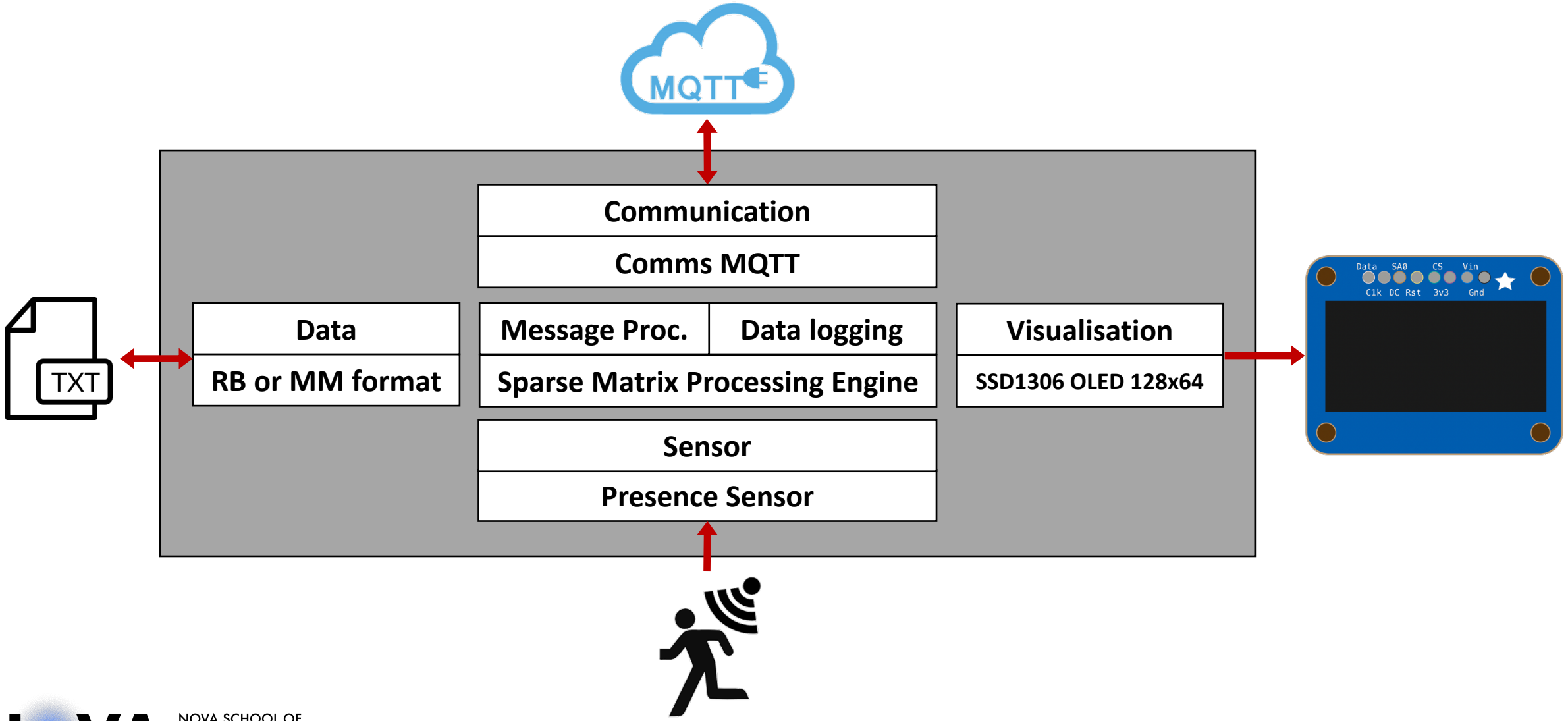
- Give purpose to the use of sparse matrixes on creating an IoT node for real-world application
- Programming Goal: **Advanced Python**
  - Group assignment (2~3 students per group)
  - Object-Oriented Programming with Python
  - Advanced software testing using unittest
- Program approach: **Test-first, 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

Real-World IoT Monitoring Scenario

Monitor the number of people entering/leaving a class room / lab at Faculty

- 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 via **MQTT** using **JSON messages** with payload using a **normalised data representation** for sparse matrices (compressed format)
- **Store & Load data locally** (in node) to prevent data loss (due to any reason)
- **Display data** (locally, at the edge) in a **visualisation dashboard** (OLED) using a pixelized **heatmap** representation for the logs

dIoTspmatrix Node: Logical Architecture



Presences’ Data Log (Sparse Matrix) & Heatmap

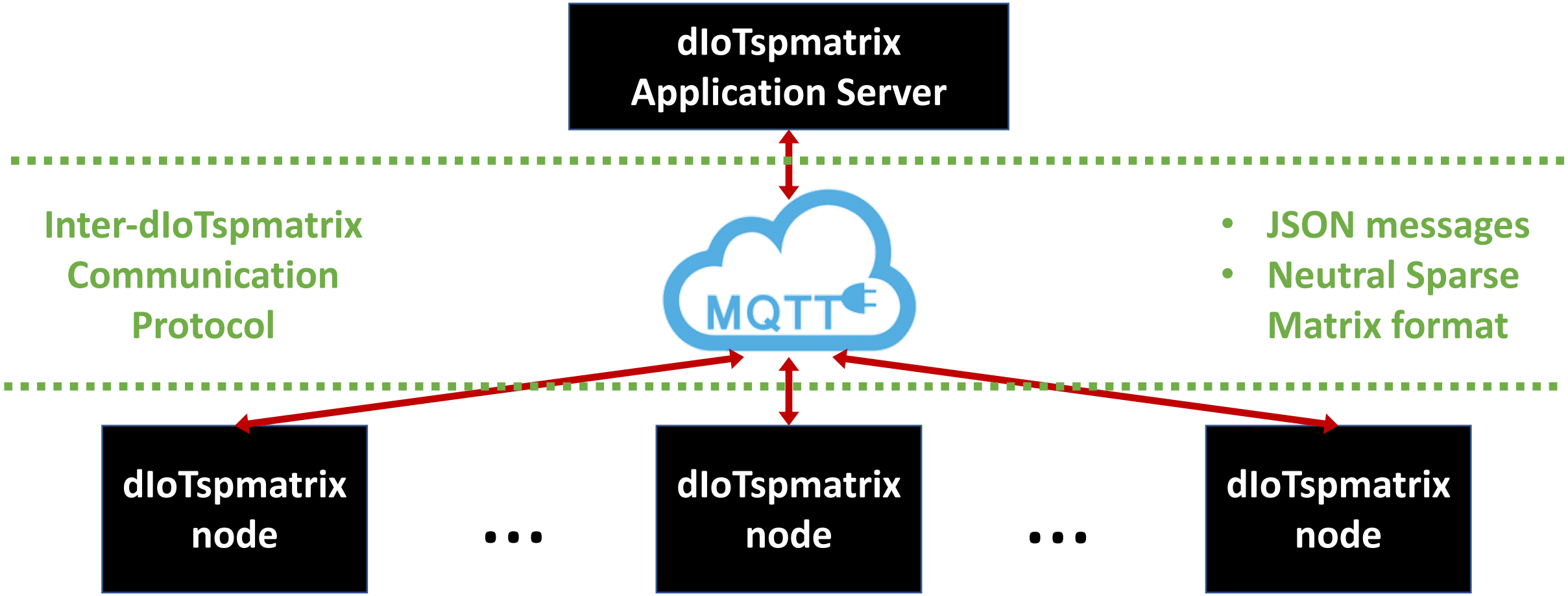
- Synthetic data for an hypotetical monday @ Lab2.1 – Ed.X, FCT NOVA

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	2	3	4	2	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	2	3	4	4	1	2	0	1	3	4	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	2	3	4	4	1	2	0	1	3	4	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	2
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

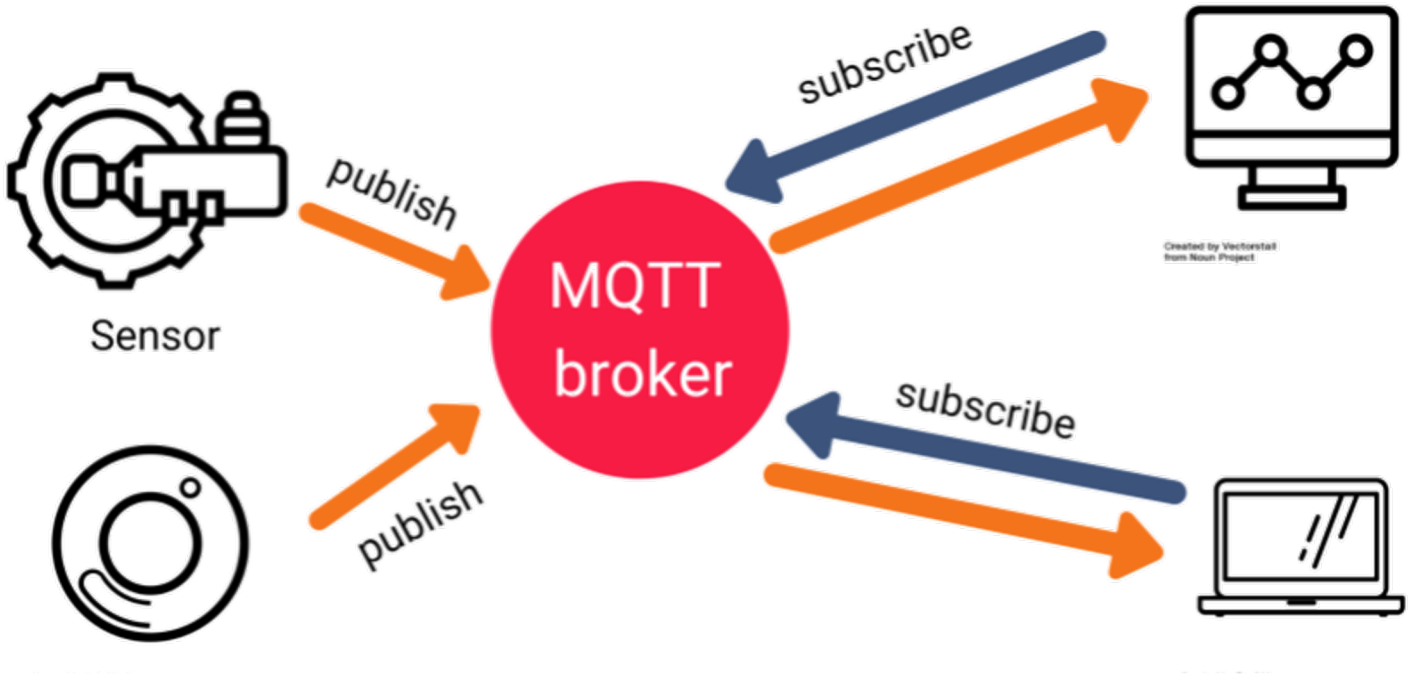
	40	41	42	42	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
14	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	2	1
16	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	2	3
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	2	1
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	3
19	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	2	0	3	2	3
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

dIoTspmatrix Communication Protocol



MQTT (Message Queuing Telemetry Transport)

- MQTT is a communication protocol for the Internet of Things (IoT) that uses the publish/subscribe communication pattern.
- The three main roles in MQTT are: Producer, Broker, and Consumer.
  - The Producer publishes a message to the Broker.
  - The Broker sends the message to the appropriate Consumer(s) that are subscribed to the specific topics.



Public MQTT Broker

- Public HiveMQ MQTT broker:
  - https://www.hivemq.com/public-mqtt-broker/
- HiveMQ MQTT Browser Client
  - http://www.hivemq.com/demos/websocket-client/
- Example Wokwi IoT Weather Station with MQTT
  - https://wokwi.com/projects/322577683855704658

  (follow the instructions in comments to run the example)

Communication Protocol JSON Messages

GET FULL LOG (24 HOURS LOG)

Request:

{ **“msg”**: <unique id>,
“request”: “get”,
“day”: 0 | -1 | -2 | ...,
“id”: “<device id>”
}

Reply:

{ **“msg”**: <unique id>,
“data”: “<sparse matrix compressed>”,
“**log_time**”: “HH:MM”
}

GET LOG OF ONE GIVEN HOUR

Request:

{ **“msg”**: <unique id>,
“request”: “get”,
“day”: 0 | -1 | -2 | ...,
“hour”: <hour>,
“id”: “<device id>” }

Reply:

{ **“msg”**: <unique id>,
“data”: “<sparse matrix compressed>”,
“**log_time**”: “HH:MM”
}

Communication Protocol JSON Messages

GET LOG OF ONE SAME MINUTE WITHIN HOURS

Request:

{ **“msg”**: <unique id>
  “request”: “get”,
  “day”: 0 | -1 | -2 | ...,
  “minute”: <minute>,
  “id”: “<device id>”
}

Reply:

{ **“msg”**: <unique id>,
  “data”: “<sparse matrix compressed>”,
  “**log_time**”: “HH:MM”
}

Communication Protocol JSON Messages

REQUEST SUM OF PRESENCES IN MANY ROOMS

Request:

{ **“msg”**: <unique id>,
“request”: “add”,
“device”: “<device id>”,
“data”: {“<sparse matrix compressed>”,
… ,
“<sparse matrix compressed>” }
}

Reply:

{ **“msg”**: <unique id>,
“data”: “<sparse matrix compressed>”
}

Let's start (programming / simulating / experimenting)

Workspace preparation:

- **Check if git is installed in your system** - Open terminal and type git – if not install, install git https://git-scm.com
- **CONFIGURE YOUR GIT IDENTITY:**
  - git config --global user.name "John Doe"
  - git config --global user.email johndoe@example.com
- **Open GitHub account** & create a **private** repository **dIoTspmatrix_[student1_number]_ _[student2_number]_[student3_number]**
- Add **Prof. Pedro Maló (Github id: pmnmalo)** as Collaborator to the new repository
- **Configure your programming environment** (VSCode, PyCharm) to the Github repository
- **MAKE SURE YOU ARE USING PYTHON 3.9 (or superior version)**

Development / Programming / Coding:

1st Step: Procedural → Object-Oriented

- Convert your Stage#1 code to Object-Oriented
  - Use the skeleton implementation provided in Moodle
- Assert that your code passes all the public tests for the Object Oriented implementation

2nd Step: Implement new sparse matrix features (TDD)

- Get the tests on the new features
- (Re)factor code to pass the tests
- Repeat until all tests provided

3rd Step: Implement the additional modules

- Implement Sensing/Logging module, Communication/Processing module, Data storage module, Visualisation module, CSR/CSC

4th Step: Simulate & Experiment

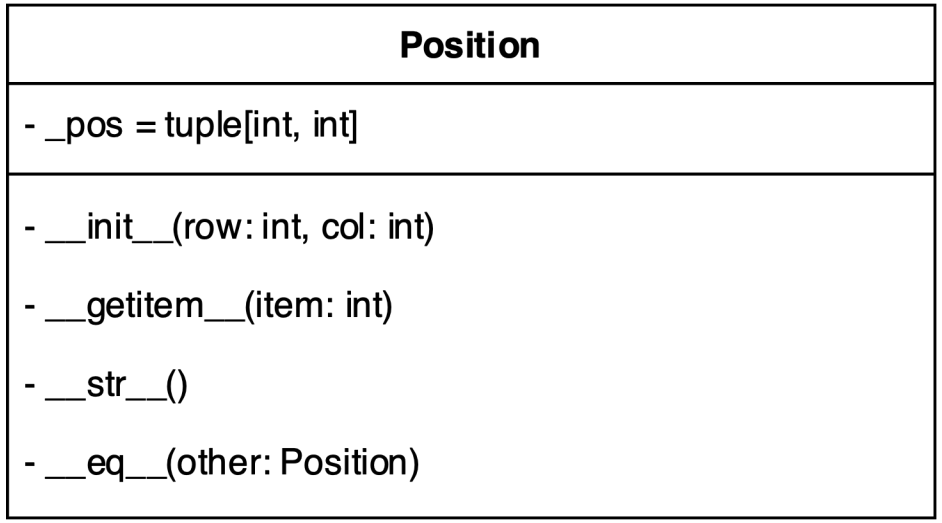
- Simulate your project with Wokwi
- Experiment your system on ESP32

PS: Document code with docstrings - https://peps.python.org/pep-0257/

1st Step: Procedural → Object-Oriented

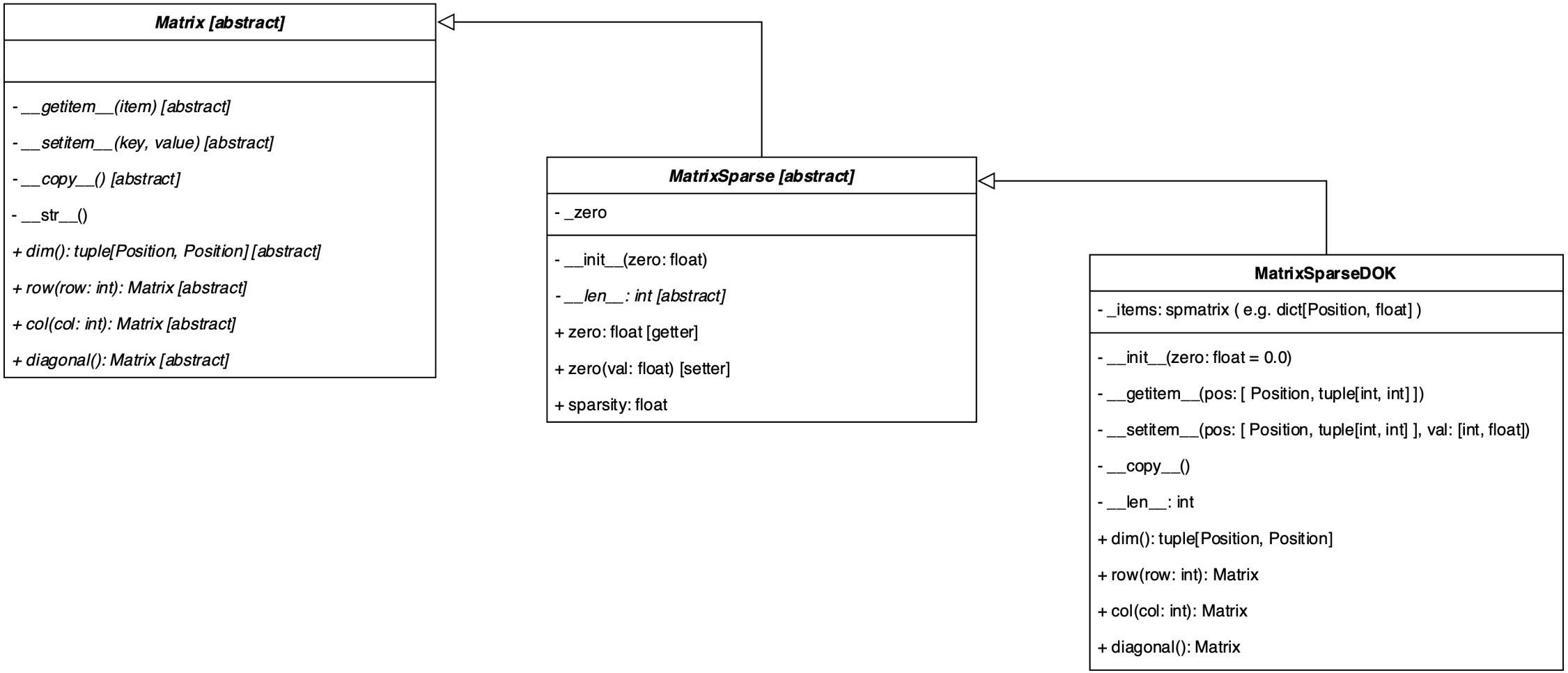
- '*_create' methods → class __init__ special method
- '*_is' methods → no longer needed, realised by object context
- 'position_row' & 'position_col' → __getitem__ special methods
- 'spmatrix_value_get' & 'spmatrix _value_set' → __getitem__ & __setitem__ sp. methods
- 'spmatrix_zero_get' & 'spmatrix_zero_set' methods → python getters & setters
- '*_copy' → __copy__ special method
- '*_str' → __str__ special method
- '*_equal' → __eq__ special method

Position: Procedural → Object-Oriented



Note: In Stage#2, we will generally set as deprecated the use of Position objects to access elements of Sparse Matrix objects – the Position concept was mainly created for having two (interlinked) modules in Stage#1 so as to demonstrate (and test) abstraction barriers to ADT structures.

Sparse Matrix: Procedural → Object-Oriented



2nd Step: New Sparse Matrix Features - Specification (1/2)

- **Iterate over the elements of the sparse matrix:**
  - Iterate only over the non-null/non-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

2nd Step: New Sparse Matrix Features - Specification (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 5-elements tuple (upper left position, zero/null value, vector of values, vector of indexes, vector of offsets)
  - If matrix is empty => raise exception ValueError with message ‘compress() empty sparse matrix’
  - If sparsity < 0.5 => raise exception ValueError with message ‘compress() dense matrix’
- **def doi(comp_vector: compressed, pos: Position) -> float**
  - Static method to directly get a value out of a sparse matrix represented using the compressed format
  - comp_vector is the 5-elements tuples sparse matrix compressed representation (out of the compress method)
    - 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**
  - Method to fully decompress a sparse matrix out of its compressed representation
  - comp_vector is the 5-elements tuples sparse matrix compressed representation (out of the compress method)
    - If comp_vector is not a compressed ADT => raise ValueError with message ‘decompress() invalid parameters’

Compress: Double-offset indexing

- **Represented as a 5-elements tuple: compressed ADT**
  - tuple[**upper_left_position**: [int, int], **zero**: float, **values**: tuple[float], **indexes**: tuple[int], **offsets**: tuple[int]]
- Compresses a sparse matrix into a representation comprised of 3 vectors:
  1. vector of **values** containing all the representable values of the sparse matrix
  2. vector of **indexes** with the indexes of the rows corresponding to each of the elements in the vector of values
  3. vector of **offsets** that indicates the starting position of each row in the two other vectors, i.e., corresponding to the offset (displacement)
- In an optimal compression, the vector of values has no null elements, so it has the length corresponding to the number of non-null elements of the sparse matrix
  - In such a case, the compressed representation looses information about what is the zero/null value of the sparse matrix and that is why the compressed 5-elements tuple ADT (out the compressed method) also includes the **zero/null value of the sparse matrix** - to make possible querying and/or reconstruction
- Also, due to the vectors being represented as lists/tuples, it is lost the information about the starting row and starting column of the matrix – due to this, the compressed 5-elements ADT includes the **upper left corner position** of the sparse matrix (as a tuple [row, col]) – to enable querying and/or reconstruction (decompress) of a sparse matrix

Double-offset indexing algorithm

- Starting with the highest density row (i.e., least sparsity), place all representable elements of that row in the vector of values starting in the first column and keeping the relative position of the elements without overlapping non-null elements. In case there is more than one row with the same density, start with the row with the lowest ordinal number.
- Fill-in the same positions in the vector of indexes with the index of that row.
- Mark the offset at the position corresponding to that row in the vector of offsets.
- Repeat for all rows in descending order of density. If it is not possible to carry out the first step, successively increment the displacement of the first column to be placed.
- At the end of the algorithm, all rows must be fully represented, i.e., all their positions, so it may be necessary to add elements with *zero/null* value (padding).
- The positions of the vector of values that have not been used are set to the *zero/null* (of the sparse matrix) and the respective vector of indexes will be filled with the value -1.

Double-offset indexing algorithm: Example #1 (optimal compression)

	1	2	3	4	5
6	0.0	0.0	6.3	0.0	0.0
7	0.0	0.0	0.0	7.4	0.0
8	8.1	8.2	0.0	0.0	8.5

Step 1:

	1	2	3	4	5
Values	8.1	8.2	.	.	8.5
Indexes	8	8	.	.	8

	6	7	8
Offsets	.	.	0

Step 2:

	1	2	3	4	5
Values	8.1	8.2	6.3	.	8.5
Indexes	8	8	6	.	8

Row	6	7	8
Offsets	0	.	0

Step 3:

	1	2	3	4	5
Values	8.1	8.2	6.3	7.4	8.5
Indexes	8	8	6	7	8

Row	6	7	8
Offsets	0	0	0

	1	2	3	4	5
Values	8.1	8.2	6.3	7.4	8.5
Indexes	8	8	6	7	8

Row	6	7	8
Offsets	0	0	0

```
(
  (6, 1),
  0.0,
  (8.1, 8.2, 6.3, 7.4, 8.5),
  (8, 8, 6, 7, 8),
  (0, 0, 0)
)
```

Double-offset indexing algorithm: Example #2 (non-optimal compression)

	1	2	3	4	5
6	0.0	6.2	6.3	0.0	0.0
7	0.0	0.0	0.0	7.4	0.0
8	8.1	8.2	0.0	0.0	8.5

Step 1:

	1	2	3	4	5
Values	8.1	8.2	.	.	8.5
Indexes	8	8	.	.	8

	6	7	8
Offsets	.	.	0

Step 2:

	1	2	3	4	5	6
Values	8.1	8.2	6.2	6.3	8.5	.
Indexes	8	8	6	6	8	.

Row	6	7	8
Offsets	1	.	0

Step 3:

	1	2	3	4	5	6	7
Values	8.1	8.2	6.2	6.3	8.5	7.4	.
Indexes	8	8	6	6	8	7	.

Row	6	7	8
Offsets	1	2	0

	1	2	3	4	5	6	7
Values	8.1	8.2	6.2	6.3	8.5	7.4	0.0
Indexes	8	8	6	6	8	7	-1

Row	6	7	8
Offsets	1	2	0

```
(
  (6, 1),
  0.0,
  (8.1, 8.2, 6.2, 6.3, 8.5, 7.4, 0.0),
  (8, 8, 6, 6, 8, 7, -1),
  (1, 2, 0)
)
```

doi : direct compressed access

def doi(comp_vector: compressed, pos: Position) -> float

- Returns the value of a given Position (input parameter *pos*) of the sparse matrix provided via its compressed representation (comp_*vector* parameter)
- If the row exists in the vector of indexes (index) and the column added to the displacement (offset) has the row number in the vector of indexes (index), return the value stored in the vector of values (value) in the same relative position. Otherwise, return the zero/null (of the sparse matrix).

	1	2	3	4	5	6	7
Values	8.1	8.2	6.2	6.3	8.5	7.4	0.0
Indexes	8	8	6	6	8	7	-1

Row	6	7	8
Offsets	1	2	0

Doi (…, Position(7,4)) = 7.4

Doi (…, Position(7,3)) = 0.0

Doi (…, Position(9,1)) = 0.0

((6, 1) , 0.0 , (8.1, 8.2, 6.2, 6.3, 8.5, 7.4, 0.0), (8, 8, 6, 6, 8, 7, -1), (1, 2, 0))

Double-offset indexing algorithm – read more

Charles N. Fischer, Ronald K. Cytron, and Richard J. LeBlanc. 2009. **Crafting a Compiler (with C)**. Addison-Wesley Publishing Company, USA.

- http://www.cs.nthu.edu.tw/~ychung/slides/CSC4180/Crafting%20a%20Compiler%20-%202010.pdf
- https://pages.cs.wisc.edu/~ansari/cc.pdf
- http://www.cs.nthu.edu.tw/~ychung/slides/CSC4180/Crafting%20a%20Compiler%20with%20C%20-%201991.pdf
- https://cpentalk.com/drive/index.php?p=Compiler+Design+Books%2FBooks%28+CPENTalk.com+%29

Evaluation Scoring : 20 points (out of 24 of choice)

- Sparse Matrix Object-Oriented Private Tests **(up to 12 points) - MANDATORY**
  - *Especially stress testing the compress, doi and decompress methods*
- Implement 1 additional Sparse Matrix representation **(up to 2 points)**
  - *Compressed Sparse Row (CSR)*
  - *Compressed Sparse Column (CSC)*
- Implement Presence Sensor Interface **(up to 2 points)**
  - *Including implementation of internal data logging capability (via interrupts)*
- Implement Communication Protocol **(up to 4 points) -- MANDATORY**
  - Ability to send & receive & process dIoTspmatrix messages
- Implement of Data import/export **(up to 2 points)**
  - *RB or MM file format*
  - *Simulated via SD Card interface in wokwi*
- Implement Visualisation Dashboard **(up to 2 points)**
  - *Dashboard with heatmap of on SSD1306 OLED 128x64 (simulated via wokwi)*