



NOVA SCHOOL OF  
SCIENCE & TECHNOLOGY

# DTSD Lab06

**dIoTspmatrix**: distributed Sparse Matrix  
processing on resource-constrained IoT devices  
v2.3

Pedro Maló ([pmm@fct.unl.pt](mailto:pmm@fct.unl.pt))

**2<sup>st</sup> stage: FULL dIoTspmatrix**  
**(for evaluation)**

# dIoTspmatrix: Development approach

## 1<sup>st</sup> 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)

## 2<sup>nd</sup> stage: FULL dIoTspmatrix (evaluation)

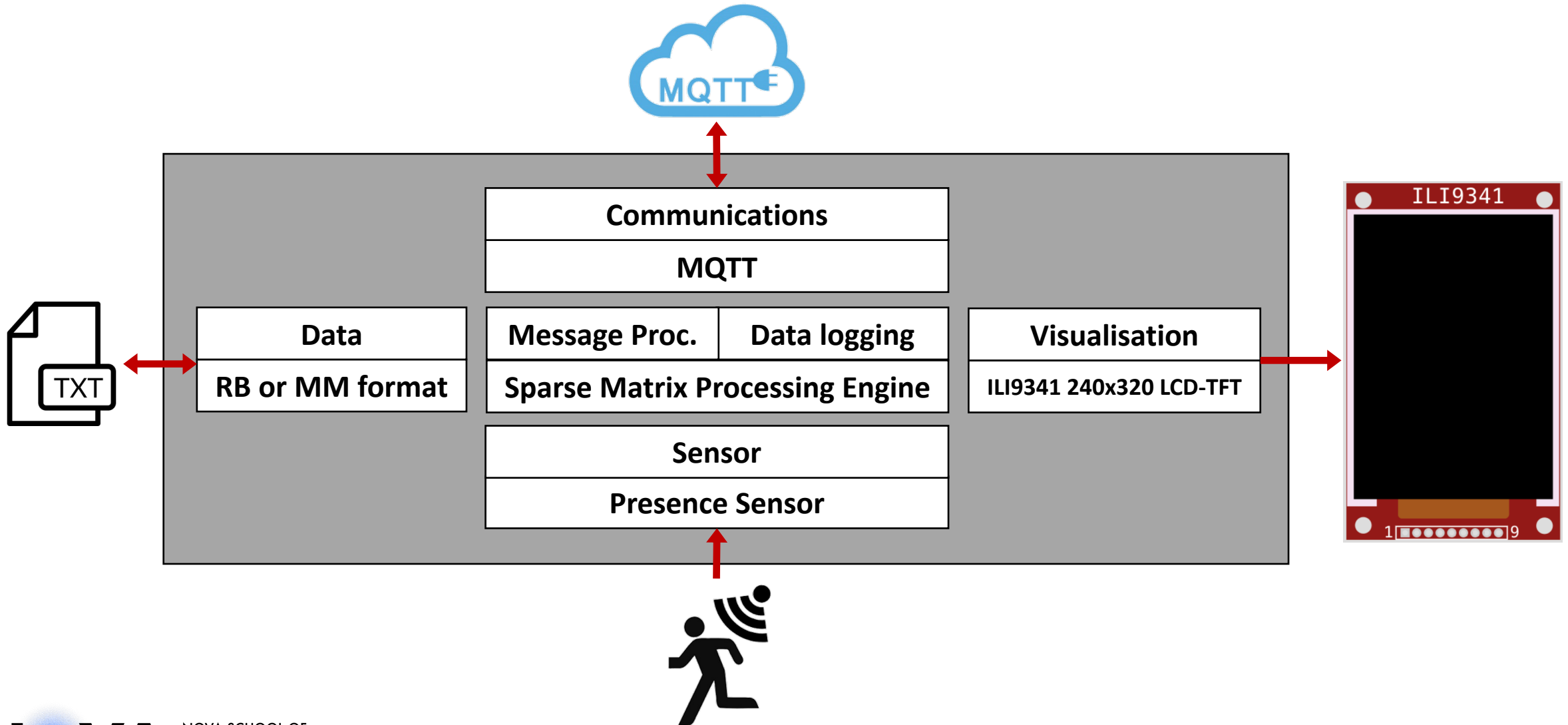
- Give purpose to the use of sparse matrixes on creating an IoT node for real-world application
- Programming Goal: Advanced Python
  - Group assignment (2~3 students per group)
  - Object-Oriented Programming with Python
  - Advanced software testing using unittest
- Program approach: Test-first, code-after
  - 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 hypothetical monday @ Lab2.1 – Ed.X, FCT NOVA

[illegible]

# **Sparse Matrix Processing Engine** (Full Features Specification)

# 1<sup>st</sup> 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 (floating point format, i.e., %g)
- ‘\*\_equal’ → `__eq__` special method

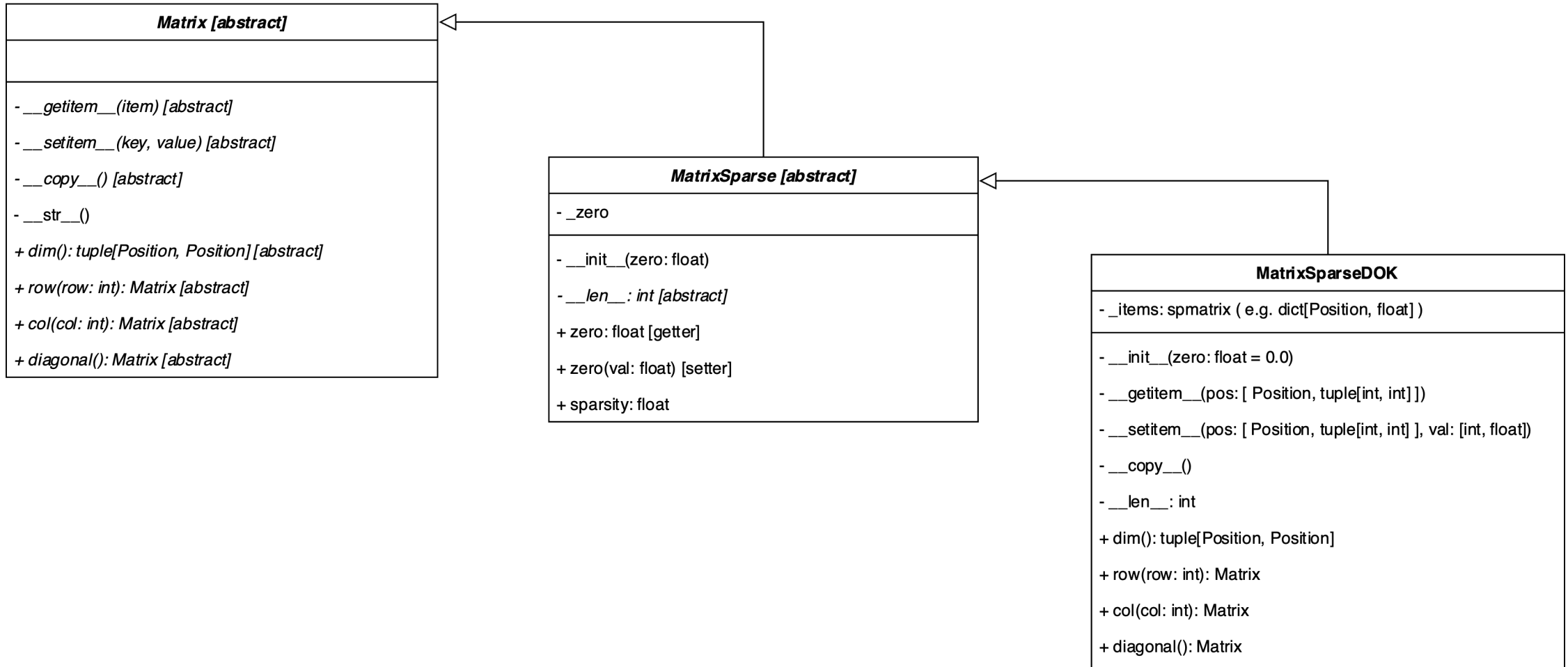


# Position: Procedural → Object-Oriented

Position
- _pos = tuple[int, int]
- __init__(row: int, col: int)
- __getitem__(item: int)
- __str__()
- __eq__(other: Position)

*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



## 2<sup>nd</sup> 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 they 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 they 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

## 2<sup>nd</sup> 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
  - If arguments are invalid => raise exception ValueError with message 'eye() invalid parameters'
- **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 the sparse matrix is empty => return (), zero, (), (), ()
  - 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 loose 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.
- 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.
- At the end of the algorithm, all elements at the end of the vector of values that are *zero/null* should be removed (truncating); naturally, the same for all elements at the end of the vector of indexes of 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



**Result:**

	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	0.0	6.4	0.0
7	0.0	0.0	7.3	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	7
Values	8.1	8.2	.	6.2	8.5	6.4	.
Indexes	8	8	.	6	8	6	.

Row	6	7	8
Offsets	2	.	0

**Step 3:**

	1	2	3	4	5	6	7	8	9
Values	8.1	8.2	.	6.2	8.5	6.4	7.3	7.4	.
Indexes	8	8	.	6	8	6	7	7	.

Row	6	7	8
Offsets	2	4	0

**Result (after truncating):**

	1	2	3	4	5	6	7	8
Values	8.1	8.2	0.0	6.2	8.5	6.4	7.3	7.4
Indexes	8	8	-1	6	8	6	7	7



Row	6	7	8
Offsets	2	4	0



(  
 (6, 1),  
 0.0,  
 (8.1, 8.2, 0.0, 6.2, 8.5, 6.4, 7.3, 7.4),  
 (8, 8, -1, 6, 8, 6, 7, 7),  
 (2, 4, 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	8
Values	8.1	8.2	0.0	6.2	8.5	6.4	7.3	7.4
Indexes	8	8	-1	6	8	6	7	7

Row	6	7	8
Offsets	2	4	0



Doi ( ..., Position(7,4) ) = 7.4

Doi ( ..., Position(8,1) ) = 8.1

Doi ( ..., Position(7,2) ) = 0.0

Doi ( ..., Position(9,1) ) = 0.0

((6, 1), 0.0, (8.1, 8.2, 0.0, 6.2, 8.5, 6.4, 7.3, 7.4), (8, 8, -1, 6, 8, 6, 7, 7), (2, 4, 0) )

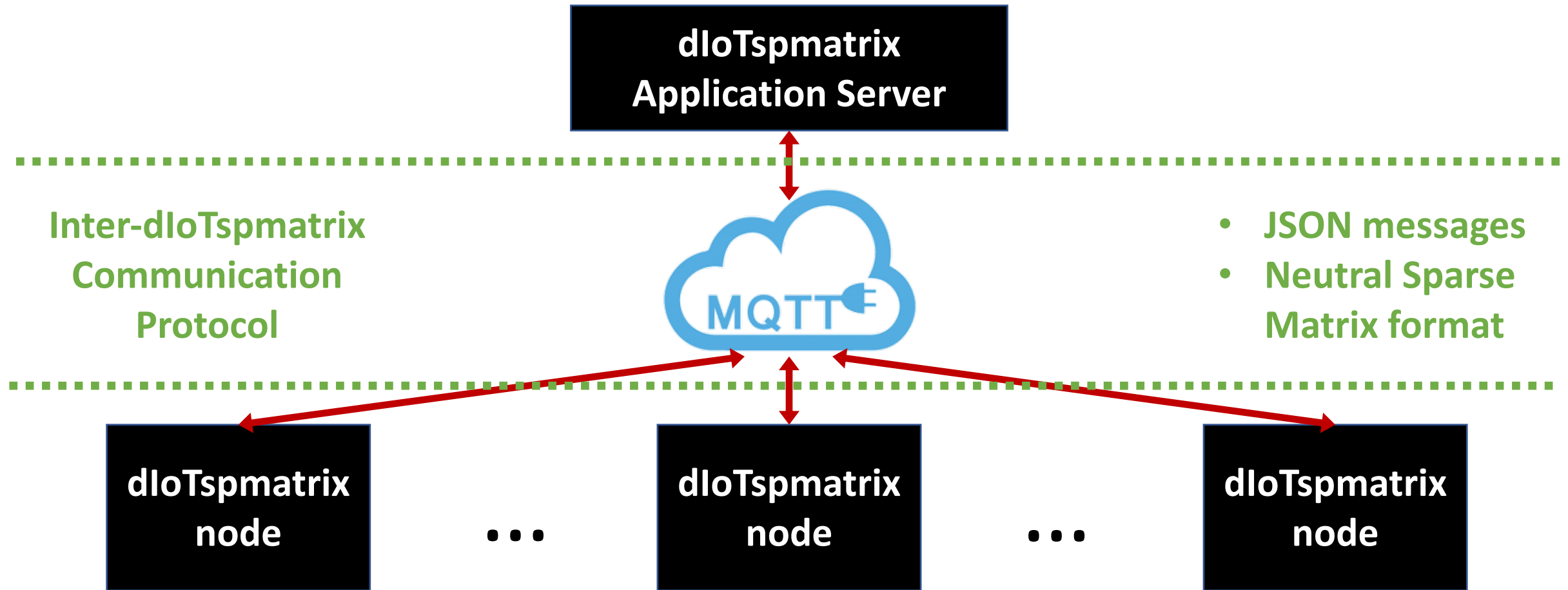
# Double-offset indexing algorithm – read more

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>

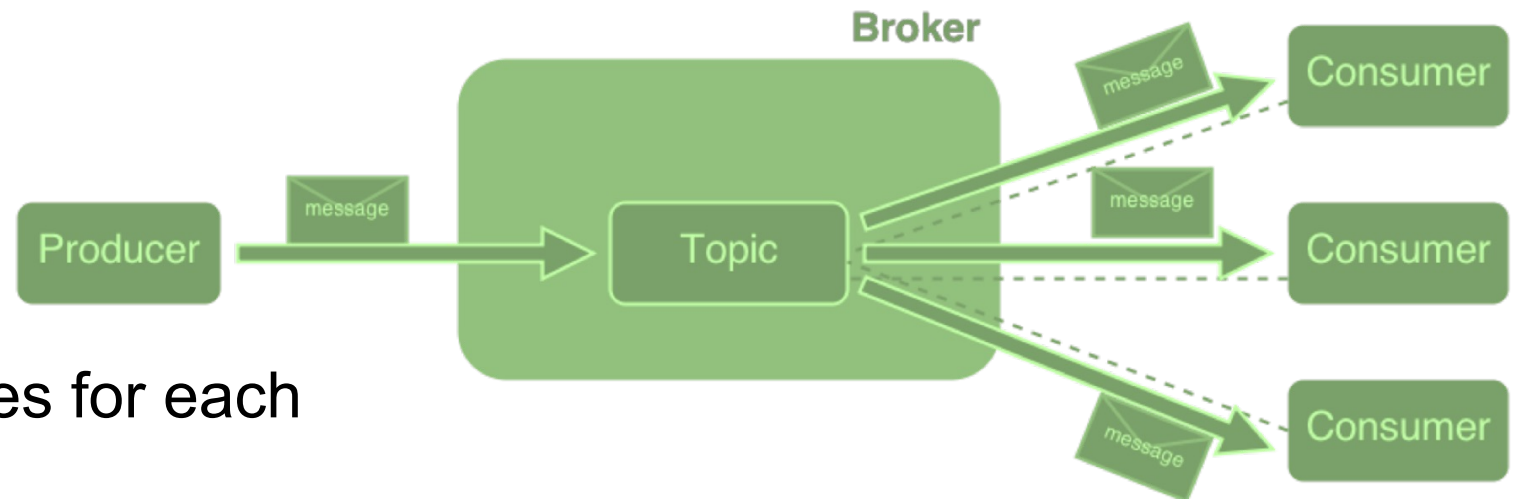
# **dIoTspmatrix Communication Protocol Specification**

# 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 **topic**.
- **Topic** is a string that the broker uses to filter messages for each connected consumer:
  - **dIoTspmatrix nodes use 3 common topics: NET, CMD, DATA**



# Communication Protocol JSON Messages [1/4]

## GET FULL LOG (24 HOURS LOG)

### Node Gets Request (out of topic CMD):

```
{ "cmd": "GET-NODE-LOG-FULL",  
  "day": 0 | -1 | -2 | ...,  
  "node_from": "<unique from node id>",  
  "node_to": "<unique to node id>" | "ANY",  
  "msg_id": "<unique message id>" }
```

### Node Publishes Reply (to topic DATA):

```
{ "cmd": "GET-NODE-LOG-FULL",  
  "data": <sparse matrix compressed>,  
  "log_time": "HH:MM",  
  "node_from": "<unique from node id>",  
  "node_to": "<unique to node id>",  
  "msg_id": "<unique message id>" }
```

The command requested to be executed by node

The ID of the node. You can name your node as your project name (that should be unique for all nodes) or else create any unique identifier.

E.g., create UUID using the device's ethernet address:

```
import machine  
id = machine.unique_id()  
print('{:02x}{:02x}{:02x}{:02x}'.format(id[0], id[1], id[2], id[3]))
```

An UUID, unique message identifier (of the request) that is to be matched by the reply message of node. To create UUIDs you can use the Python/Micropython uuid package implementation:

<https://github.com/pfalcon/pycopy-lib/blob/master/uuid/uuid.py>

# Communication Protocol JSON Messages [2/4]

## GET LOG OF ONE GIVEN HOUR

**Node Gets Request (out of topic CMD):**

```
{ "cmd": "GET-NODE-LOG-BY-HOUR",  
  "day": 0 | -1 | -2 | ...,  
  "hour": <hour>,  
  "node_from": "<unique from node id>",  
  "node_to": "<unique to node id>" | "ANY",  
  "msg_id": "<unique message id>" }
```

**Node Publishes Reply (to topic DATA):**

```
{ "cmd": "GET-NODE-LOG-BY-HOUR",  
  "data": <sparse matrix compressed>,  
  "log_time": "HH:MM",  
  "node_from": "<unique from node id>",  
  "node_to": "<unique to node id>",  
  "msg_id": "<unique message id>" }
```

## GET LOG OF ONE SAME MINUTE WITHIN HOURS

**Node Gets Request (out of topic CMD):**

```
{ "cmd": "GET-NODE-LOG-BY-MINUTE",  
  "day": 0 | -1 | -2 | ...,  
  "minute": <minute>,  
  "node_from": "<unique from node id>",  
  "node_to": "<unique to node id>" | "ANY",  
  "msg_id": "<unique message id>" }
```

**Node Publishes Reply (to topic DATA):**

```
{ "cmd": "GET-NODE-LOG-BY-MINUTE",  
  "data": <sparse matrix compressed>,  
  "log_time": "HH:MM",  
  "node_from": "<unique from node id>",  
  "node_to": "<unique to node id>",  
  "msg_id": "<unique message id>" }
```

# Communication Protocol JSON Messages [3/4]

## GET SUM FULL LOG FROM ALL NODES

### Node Gets Request (out of topic CMD):

```
{ "cmd": "GET-LOG-FULL-EDGE",  
  "day": 0 | -1 | -2 | ...,  
  "node_from": "<unique from node id>",  
  "node_to": "<unique to node id>" | "ANY",  
  "msg_id": "<unique message id>" }
```

### Node Publishes Reply (to topic DATA):

```
{ "cmd": "GET-LOG-FULL-EDGE",  
  "data": <sparse matrix compressed>,  
  "log_time": "HH:MM",  
  "node_from": "<unique from node id>",  
  "node_to": "<unique to node id>",  
  "msg_id": "<unique message id>" }
```

- Gets the sum of the full logs of all the nodes in the logging network
- The requested node needs to get itself the needed data from all other nodes on their full logs
- Then sums (edge computing) all the data collected from other nodes
  - Accounting of possibly different log\_time received from other nodes
- Then, publish the sum of the logs



# Communication Protocol JSON Messages [4/4]

## NODE ALIVE MESSAGE

**Node Publishes (to topic NET):**

```
{ "cmd": "NODE ALIVE",  
  "node_from": "unique from node id>",  
  "timestamp": "YYYY-MM-DD HH:MM:SS"  
}
```

- On connecting to the logging network and once each minute, the node sends a 'keep alive' message (to the 'NET' topic) to let all other know that is active in the data logging network.

## NODE DEAD MESSAGE

**Node Publishes (to topic NET):**

```
{ "cmd": "NODE DEAD",  
  "node_from": "unique from node id>",  
  "timestamp": "YYYY-MM-DD HH:MM:SS"  
}
```

- On disconnecting from the logging network, a node can publish a 'node dead' message (to the 'NET' topic) to let all other nodes know that is now inactive in the data logging network.

# Simulating / Testing in Wokwi (using Micropython on ESP32)

# dloTspmatrix project in wokwi

- Start a New Project in Wokwi of type: MicroPython on ESP32
- Create Files Position.py, Matrix.py, MatrixSparse.py, MatrixSparseDOK.py
- Copy source code from your project (GitHub, PyCharm, VSCode) to Wokwi
- Delete imports to `__future__` - Micropython does not support Future statement definitions
- Delete all elements related to Abstract Classes/Methods from Wokwi code
  - Micropython in Wokwi still does not support the Abstract Based Classes (PEP 3119)
  - Delete imports from `abc`
    - Delete **"from abc import ABC, abstractmethod"**
  - Remove inheritance of class Matrix from ABC
    - Delete **class** Matrix(ABC) → **class** Matrix()
  - Delete **@abstractmethod** decorator from the designated abstract methods
- Remove all type aliases definitions from code – Micropython does not support type aliases
  - Delete type aliases definitions and replace (or simply remove) the related type aliases references in the code
    - Delete type alias: `position = tuple[int, int]`, `compressed = tuple[[position,()], float, tuple[float,...], tuple[int,...], tuple[int,...]]`
    - Delete type alias: `_zero = float` | `matrix = dict[Position, float]` | `_items = matrix`
- If you get "TypeError: object of type 'dict\_view' has no len()" convert the dict\_view to list with `list()`

# Communication Protocol Test on wokwi

```
"""
```

```
dbIoTspmatrix MQTT message processing wokwi demo
```

```
To view the data:
```

1. Go to <http://www.hivemq.com/demos/websocket-client/>
2. Click "Connect"
3. Under Subscriptions, click "Add New Topic Subscription"
4. In the Topic field, type "dIoTspmatrix-data" then click "Subscribe"
5. In the Topic field, type "dIoTspmatrix-cmd" then click "Subscribe"

```
"""
```

```
from MatrixSparseDOK import *
```

```
import machine
import network
import time
import ujson
from uuid import *
from umqtt.simple import MQTTClient
```

```
# MQTT Server Parameters
```

```
id = machine.unique_id()
MQTT_CLIENT_ID = '{:02x}{:02x}{:02x}{:02x}'.format(id[0], id[1], id[2], id[3])
MQTT_BROKER = "broker.mqttdashboard.com"
MQTT_USER = ""
MQTT_PASSWORD = ""
MQTT_TOPIC_DATA = "dIoTspmatrix-data"
MQTT_TOPIC_CMD = "dIoTspmatrix-cmd"
```

```
# Connect to WiFi
```

```
print("Connecting to WiFi", end="")
sta_if = network.WLAN(network.STA_IF)
sta_if.active(True)
sta_if.connect('Wokwi-GUEST', '')
while not sta_if.isconnected():
    print(".", end="")
    time.sleep(0.1)
print(" Connected!")
```

```
# Connect to MQTT server
```

```
print("Connecting to MQTT server... ", end="")
client = MQTTClient(MQTT_CLIENT_ID, MQTT_BROKER, user=MQTT_USER, password=MQTT_PASSWORD)
client.connect()
print("Connected!")
```

```
cmd_messages = list()
```

```
def mqtt_callback(topic, message):
```

```
    global cmd_messages
```

```
    try:
```

```
        cmd_msg = ujson.loads(message)
```

```
        if cmd_msg["node_to"] == MQTT_CLIENT_ID or cmd_msg["node_to"] == "ANY":
            cmd_messages.append(cmd_msg)
```

```
    except:
```

```
        pass
```

```
client.set_callback(mqtt_callback)
```

```
client.subscribe(MQTT_TOPIC_CMD)
```

```
m1 = MatrixSparseDOK()
```

```
m1_data = {(5, 5): 5.5, (5, 6): 5.6, (6, 7): 6.7, (7, 4): 7.4, (7, 5): 7.5, (7, 8): 7.8}
```

```
for key, value in m1_data.items():
```

```
    m1[Position(key[0], key[1])] = value
```

```
vc = m1.compress()
```

```
def process_commands():
```

```
    global cmd_messages
```

```
    for i in range(len(cmd_messages)):
```

```
        cmd_msg = cmd_messages[0]
```

```
        print(cmd_msg)
```

```
        if cmd_msg["cmd"] == "GET-NODE-LOG-FULL":
```

```
            message = ujson.dumps({
```

```
                "cmd": "GET-NODE-LOG-FULL",
```

```
                "node_from": MQTT_CLIENT_ID,
```

```
                "node_to": cmd_msg["node_from"],
```

```
                "data": vc,
```

```
                "msg_id": cmd_msg["msg_id"],
```

```
                "log_time": "23:59"
```

```
            })
```

```
            client.publish(MQTT_TOPIC_DATA, message)
```

```
            cmd_messages.pop(0)
```

```
# Loop forever checking and processing MQTT messages
```

```
while (True):
```

```
    client.check_msg()
```

```
    process_commands()
```

```
    time.sleep(1)
```

# Communication Protocol JSON Test Messages

```
{ "cmd": "GET-NODE-LOG-FULL",  
  "day": 0,  
  "node_from": "240ac400",  
  "node_to": "ANY",  
  "msg_id": "7d20da9a-95e0-4165-b56d-90e09a0127f4"  
}
```

```
{ "cmd": "GET-NODE-LOG-FULL",  
  "data": [[5, 4], 0.0, [7.4, 7.5, 5.5, 5.6, 7.8, 6.7, 0.0], [7, 7, 5, 5, 7, 6, -1], [1, 2, 0]],  
  "log_time": "23:59",  
  "node_from": "240ac400",  
  "node_to": "240ac400",  
  "msg_id": "7d20da9a-95e0-4165-b56d-90e09a0127f4"  
}
```

# 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)

# Evaluation (Scoring)

# 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 dloTspmtrix 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 ILI9341 240x320 LCD-TFT (simulated via wokwi)*



**Let's Go !**

# 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 **dloTspmatriX\_[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:

### 1<sup>st</sup> 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

### 2<sup>nd</sup> 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

### 3<sup>rd</sup> Step: Implement the additional modules

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

### 4<sup>th</sup> Step: Simulate & Experiment

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

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