Machine Learning Core with STM MEMS Sensor LSM6DSOX

This report demonstrates how to use the LSM6DSOX Machine Learning Sensor in combination with the B-L072Z-LRWAN1 and the X-NUCLEO-IKS01A3, create a Decision Tree with Weka and use it to get feedback from the actions performed.

Hardware that was used in this project:

Board	Extender	Adapter
B-L072Z-LRWAN1	X-NUCLEO-IKS01A3	MKI197V1 (LSM6DSOX)

- *Getting the Firmware and Software needed*:

The **Firmware** that was used is the X-CUBE-MEMS1 Firmware Package.

Note: Inside the repository folder **Projects**, the only available options are Nucleo-based projects. The ones I tested though, work just fine with other ST boards, since we are using one of the **Extenders** that are compatible: (**IKS01A1/IKS01A2/IKS01A3**).

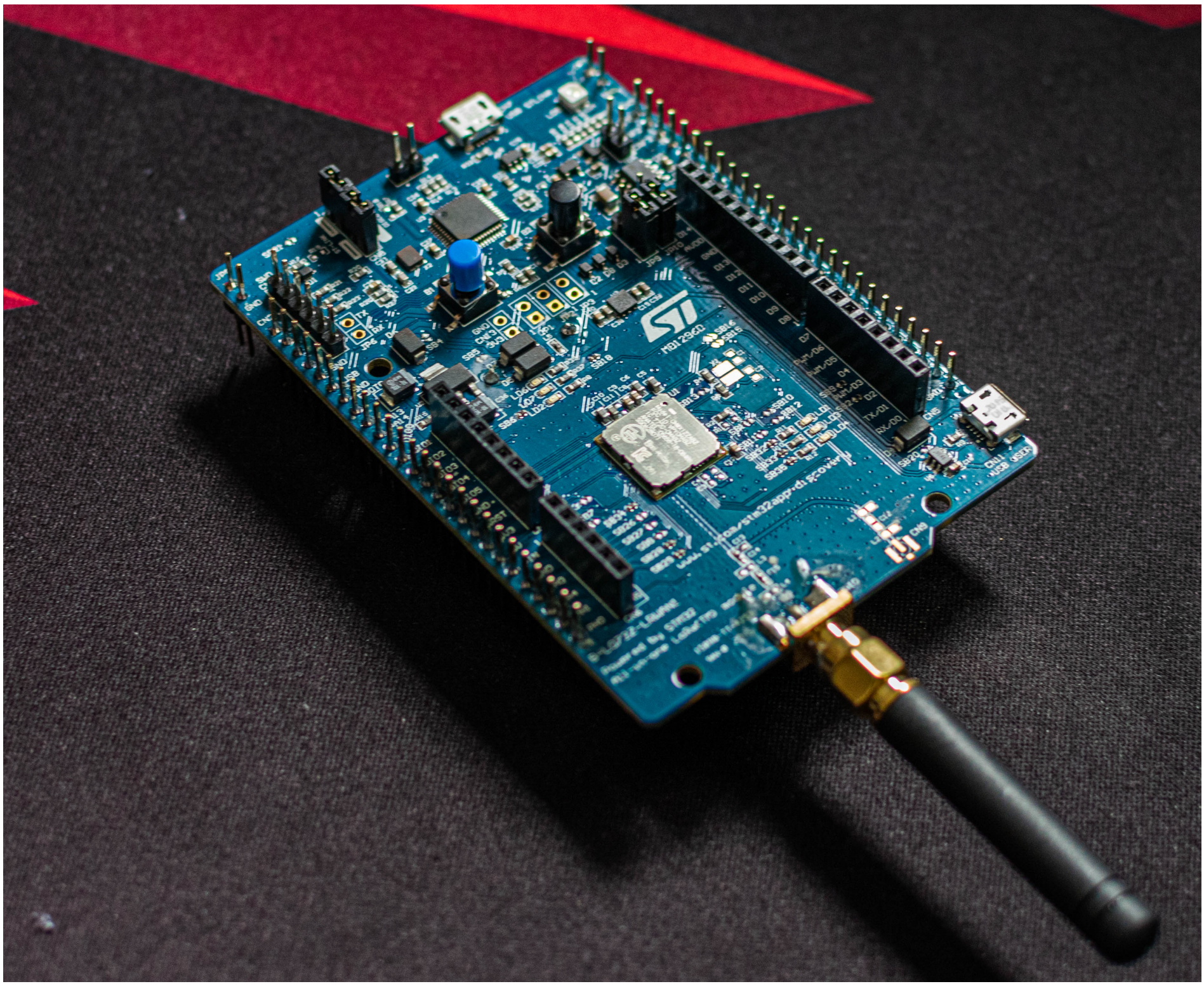
The project linked above needs to be programmed and flashed with the STM32CubeIDE program, since it does not include a CubeMX project.

The **Software** used is:

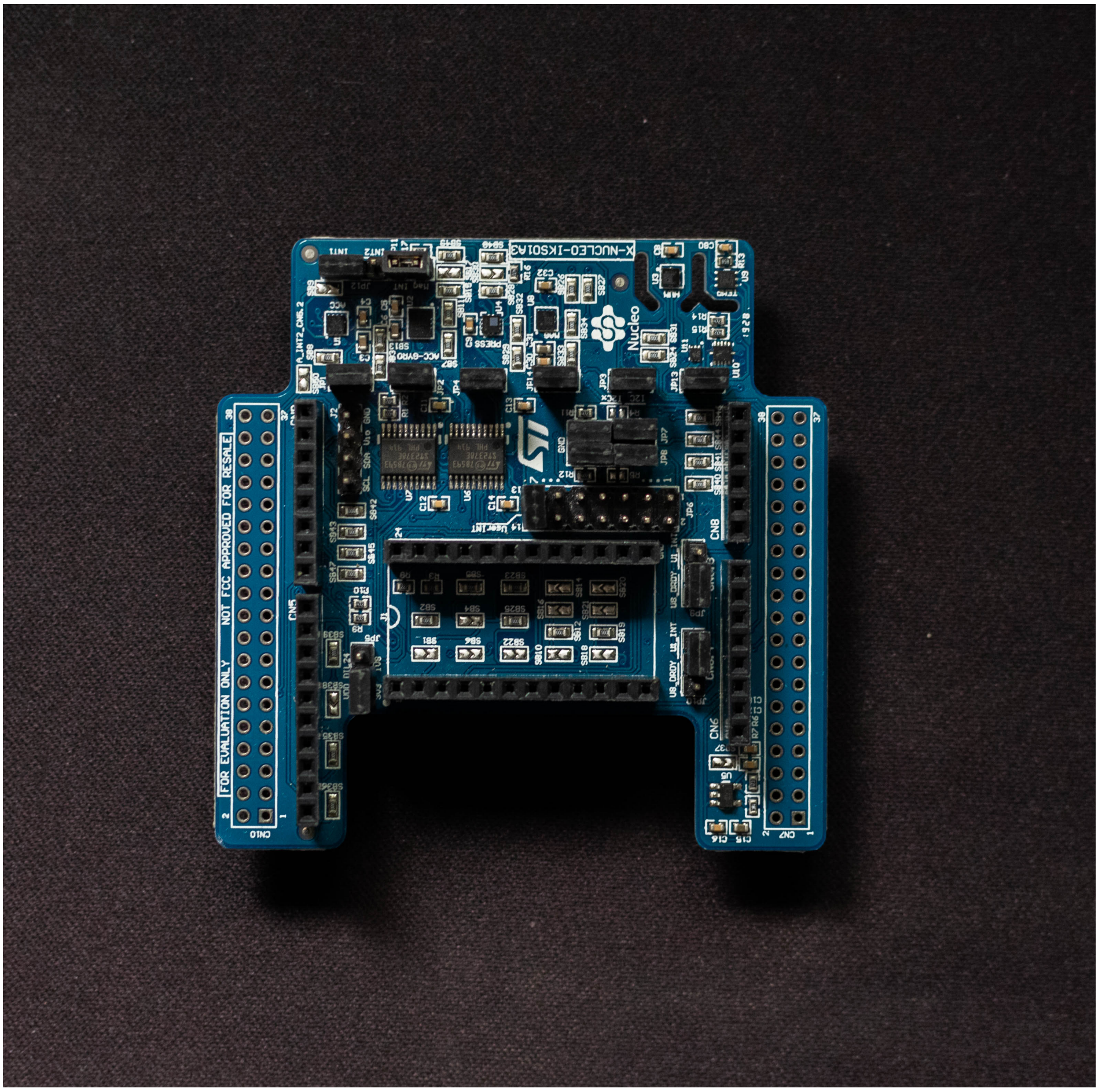
1. **Unicleo-GUI**: Demonstrates the functionality of ST sensors and algorithms
2. **Unico**: Needed to create the .ucf file that will be loaded after the Decision Tree is created.
3. **Weka**: Machine Learning Tool used for Data Mining tasks.

Hardware Combination and Specifications.

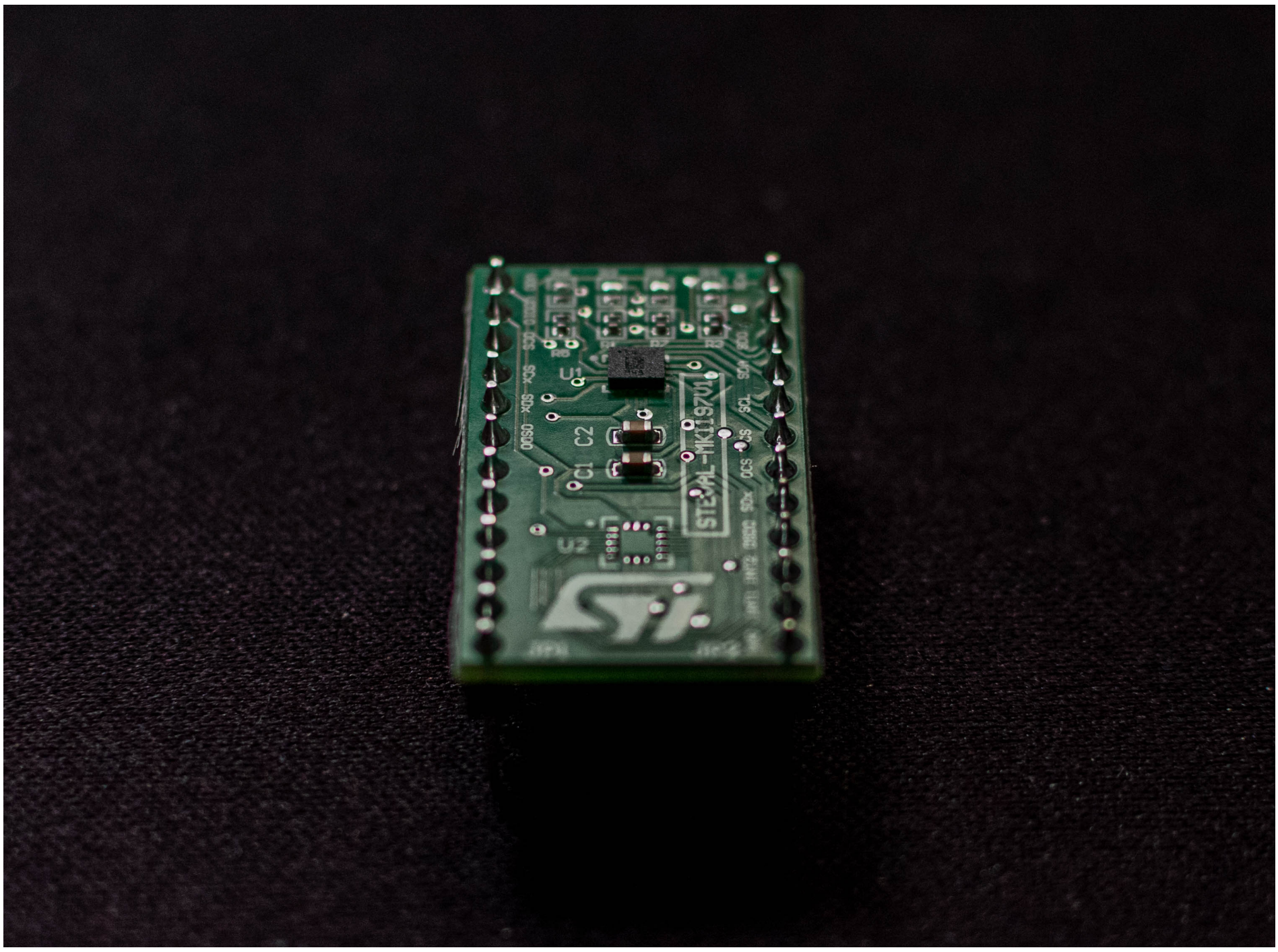
1. **B-L072Z-LRWAN1**



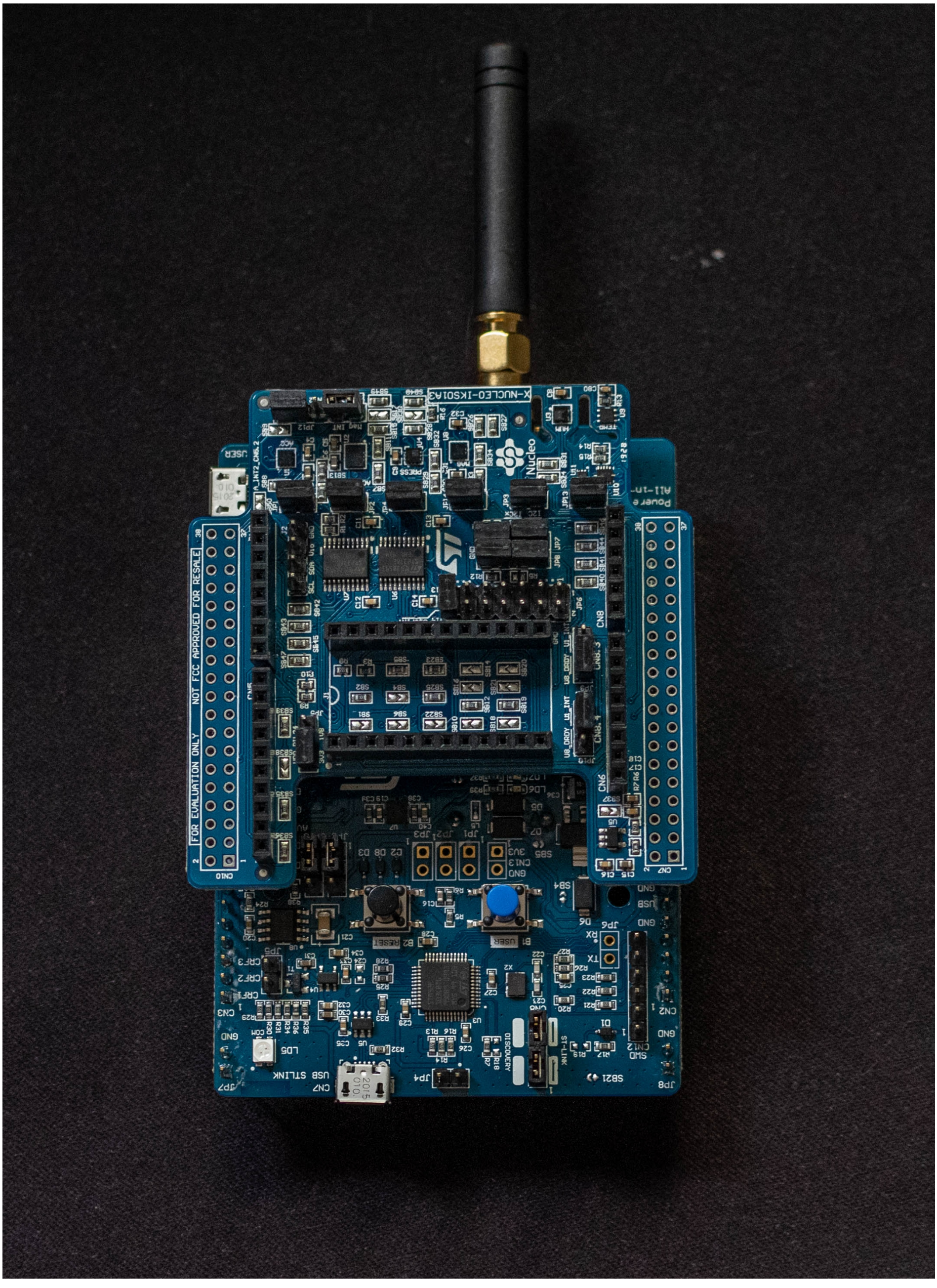
2. **X-NUCLEO-IKS01A3**



3. **MKI197V1 (LSM6DSOX)**

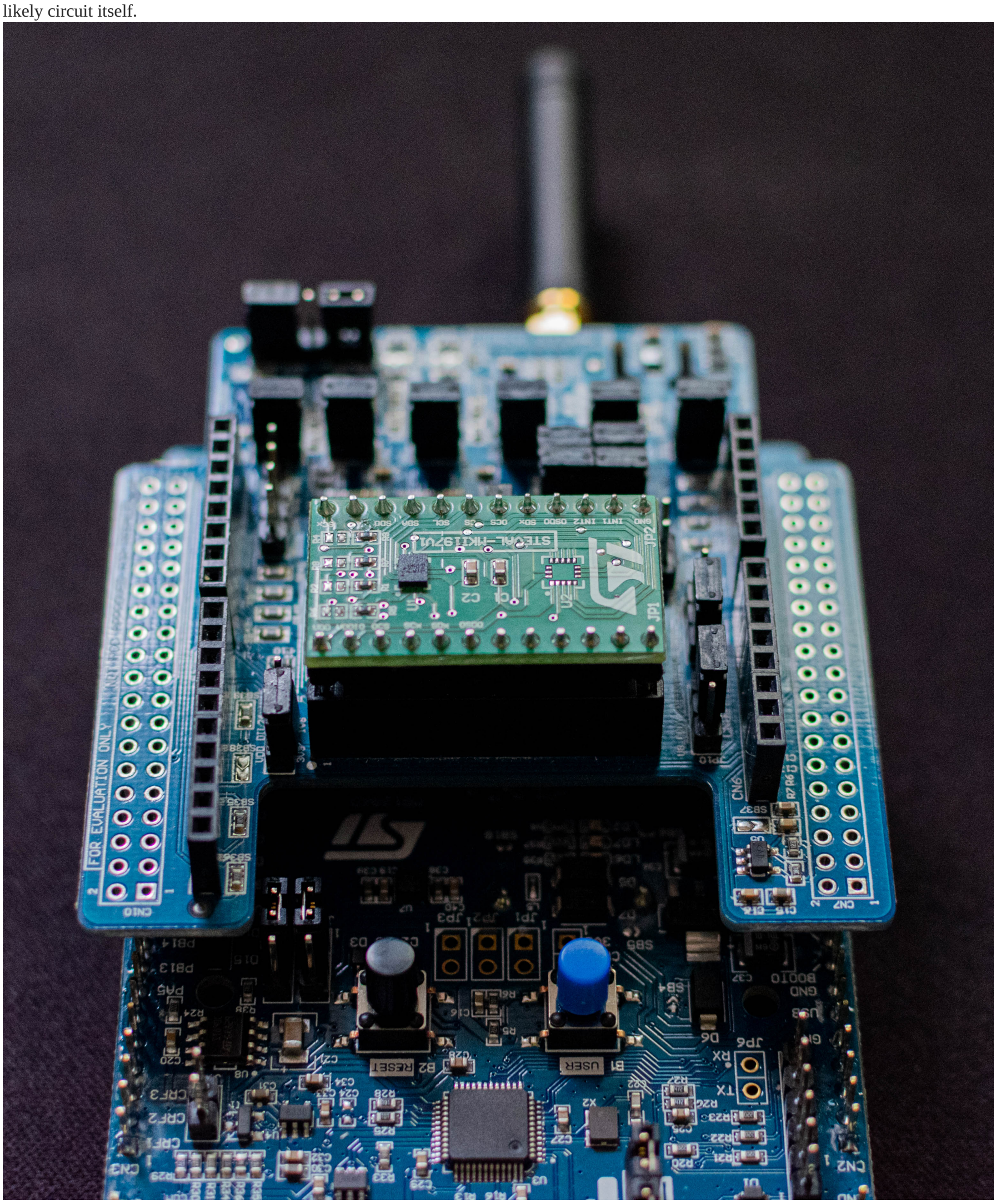


- **First we need to combine the X-NUCLEO-IKS01A3 on the Arduino R3 pinouts of B-L072Z-LRWAN1**

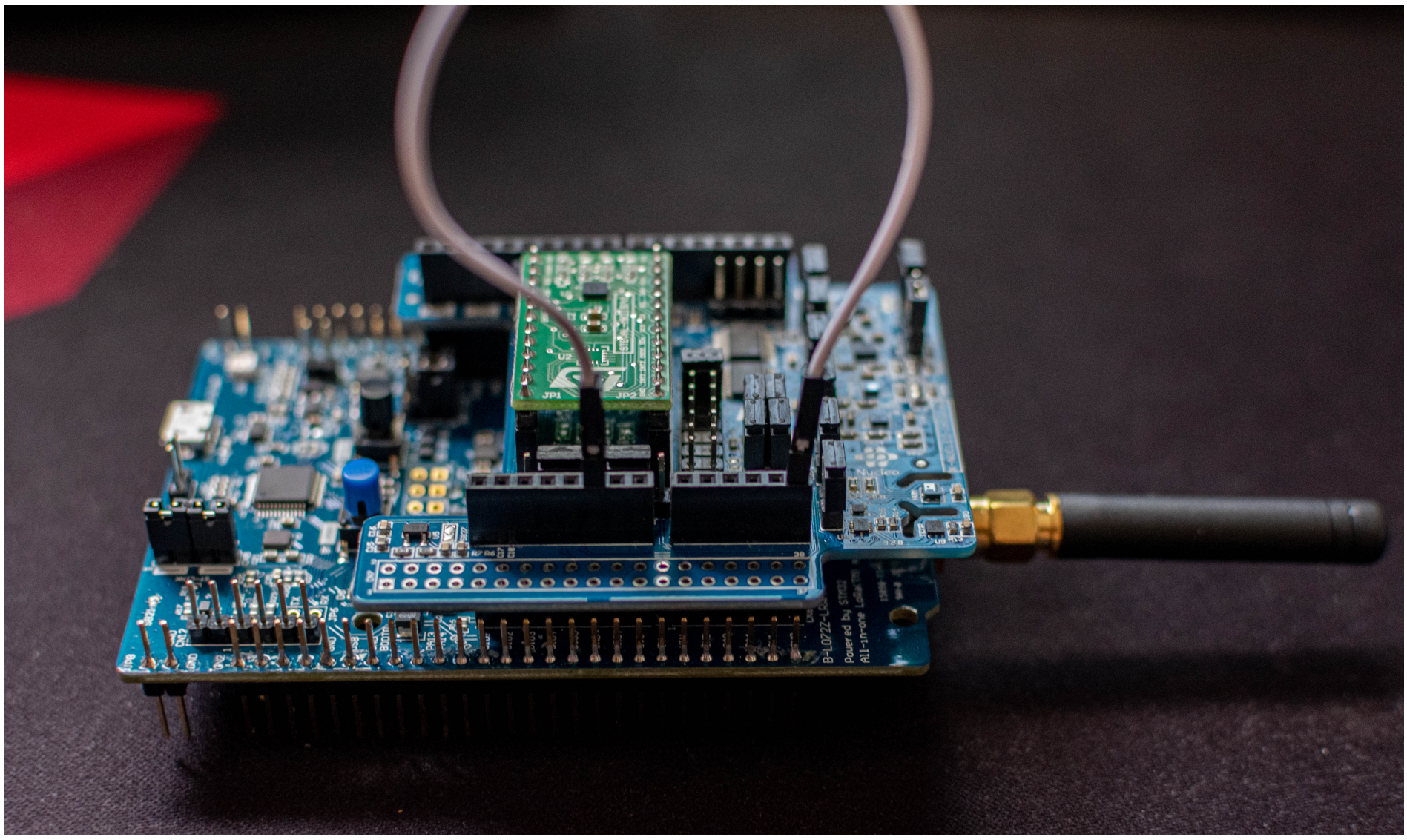


- **Next we add the MKI197V1 on the DIL24 Socket of the IKS01A3**

IMPORTANT: Make sure the ST Logo on BOTH the extender and the adapter are aligned, otherwise the adapter will overheat and most likely circuit itself.

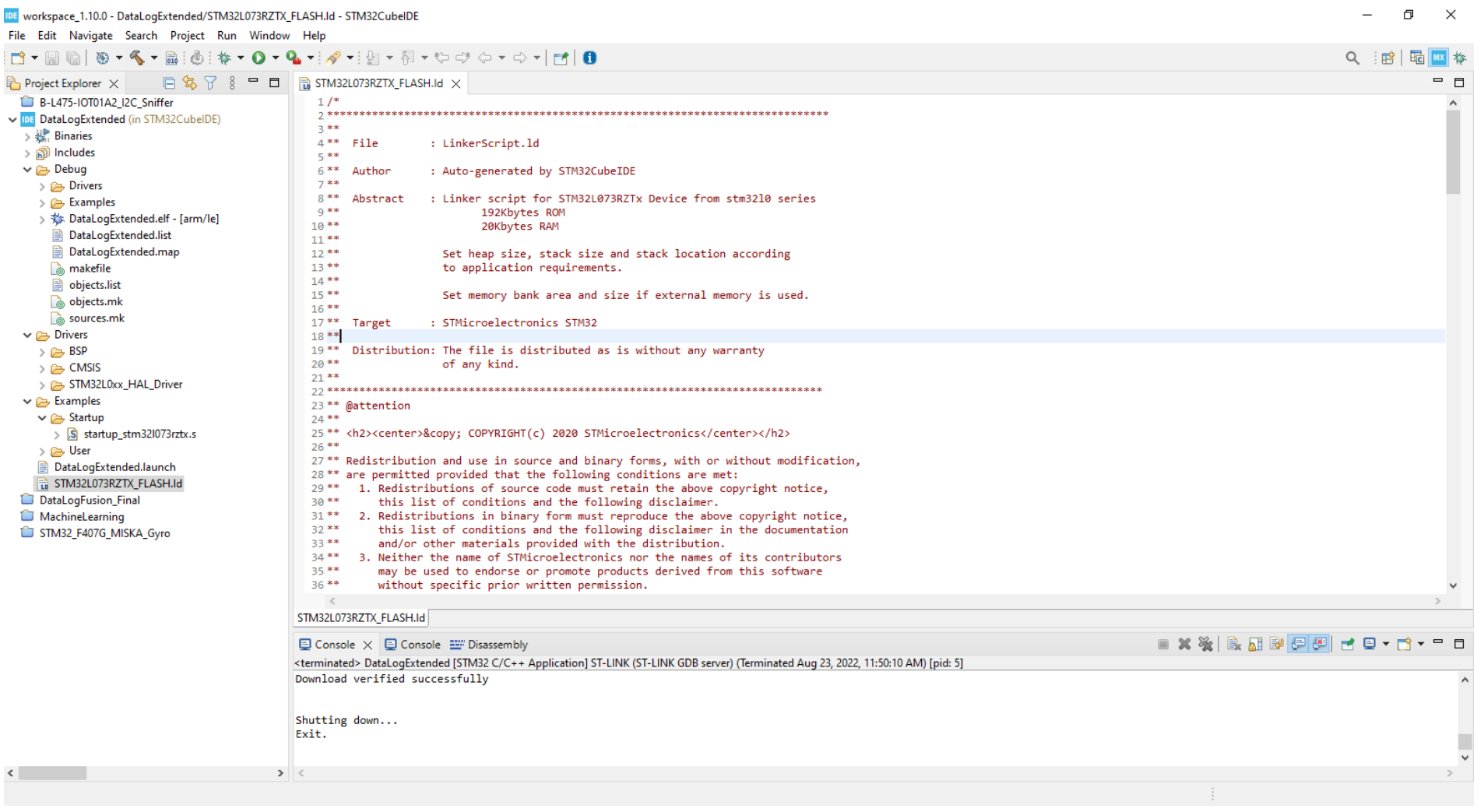


- **Now for the final step**, we need to enable the LSM6DSOX sensor on the MKI197V1. The LSM6DSOX sensor starts in I3C mode because of a level shifter on the IKS01A3 that keeps the INT1 of the LSM6DSOX high, this results to I3C initialization by default (as described in the Datasheet). The only solution that I found was to bypass the INT1 and route the INT2 in its place. That can be done by connecting the A5 pin of the IKS01A3 to GND with a wire and also change the JP6 Jumper from the default 5-6 to 13-14. The change of the Jumper supposedly sets the M_INT2_0 on pin D2, in case a change need to be made in the schematic. After these changes, the LSM6DSOX should be enabled.

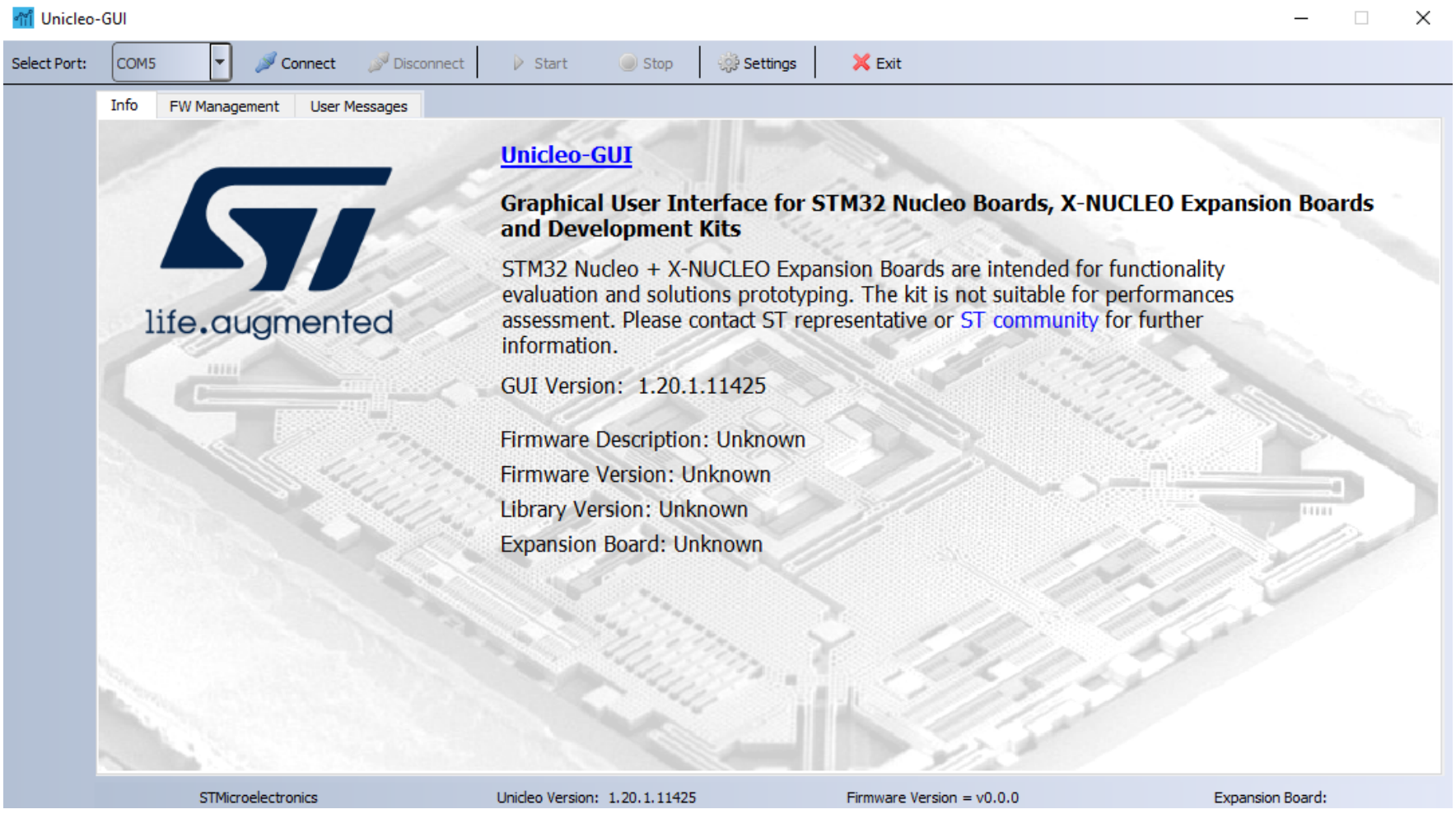


Now we are ready to begin.

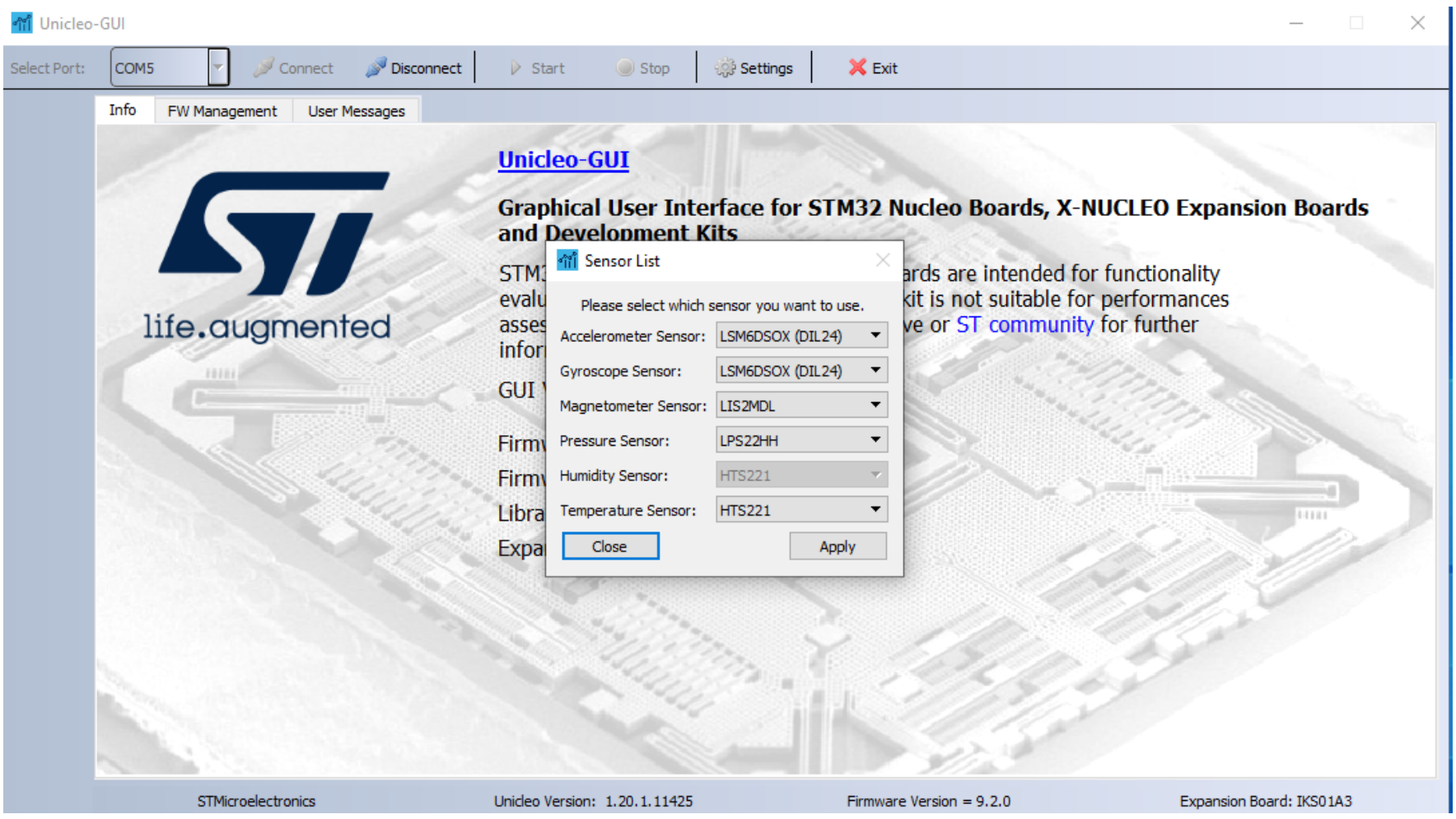
- First we need to program and flash our board. This needs to be done with STM32CubeIDE.



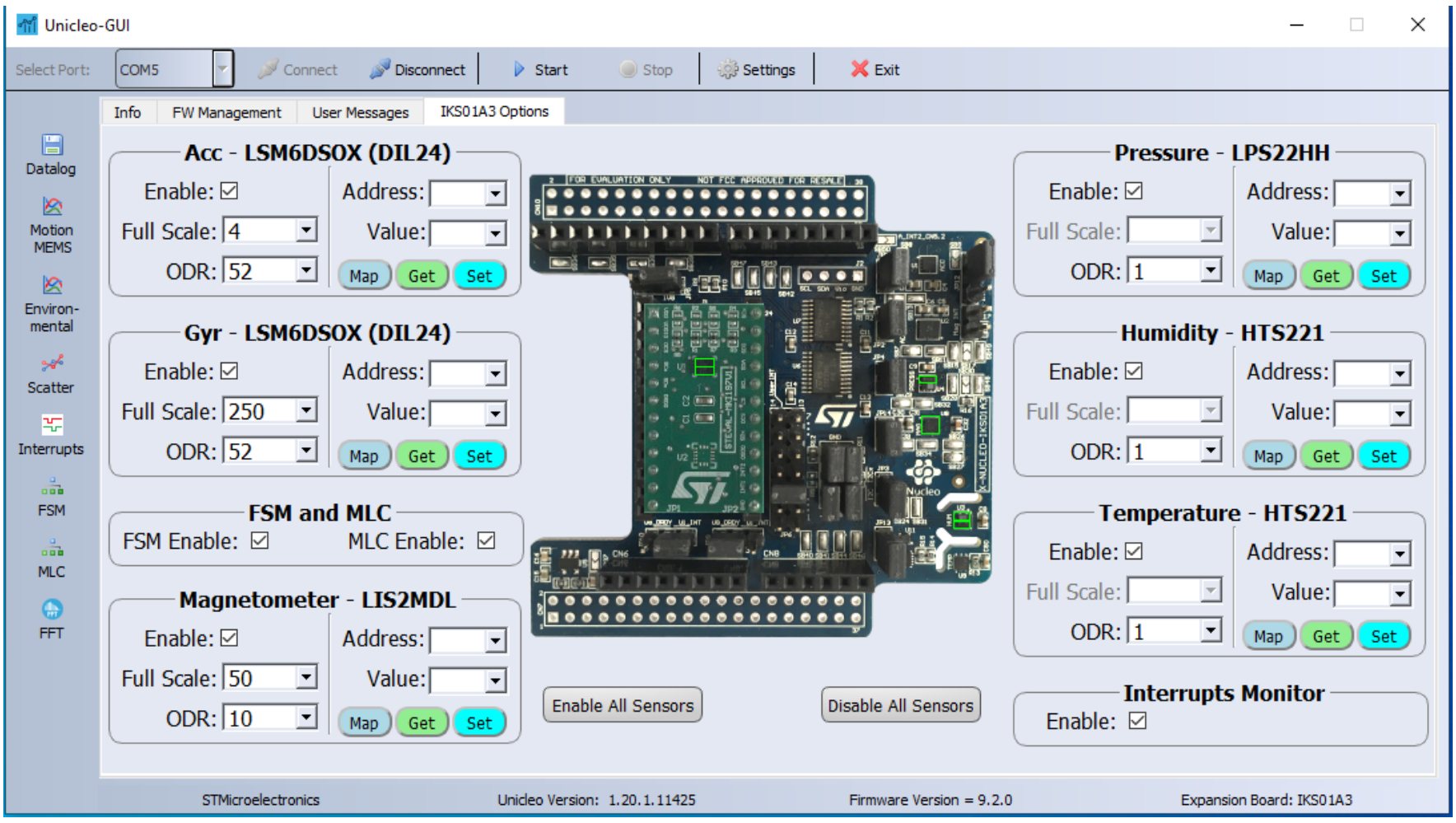
- Now let's check if our sensor works as intented in Unicleo.
    - This is the interface of Unicleo. If our board is connected via USB (ST Link) then the Serial Port should be automatically selected, mine for example is COM5. *We select* `Connect`.



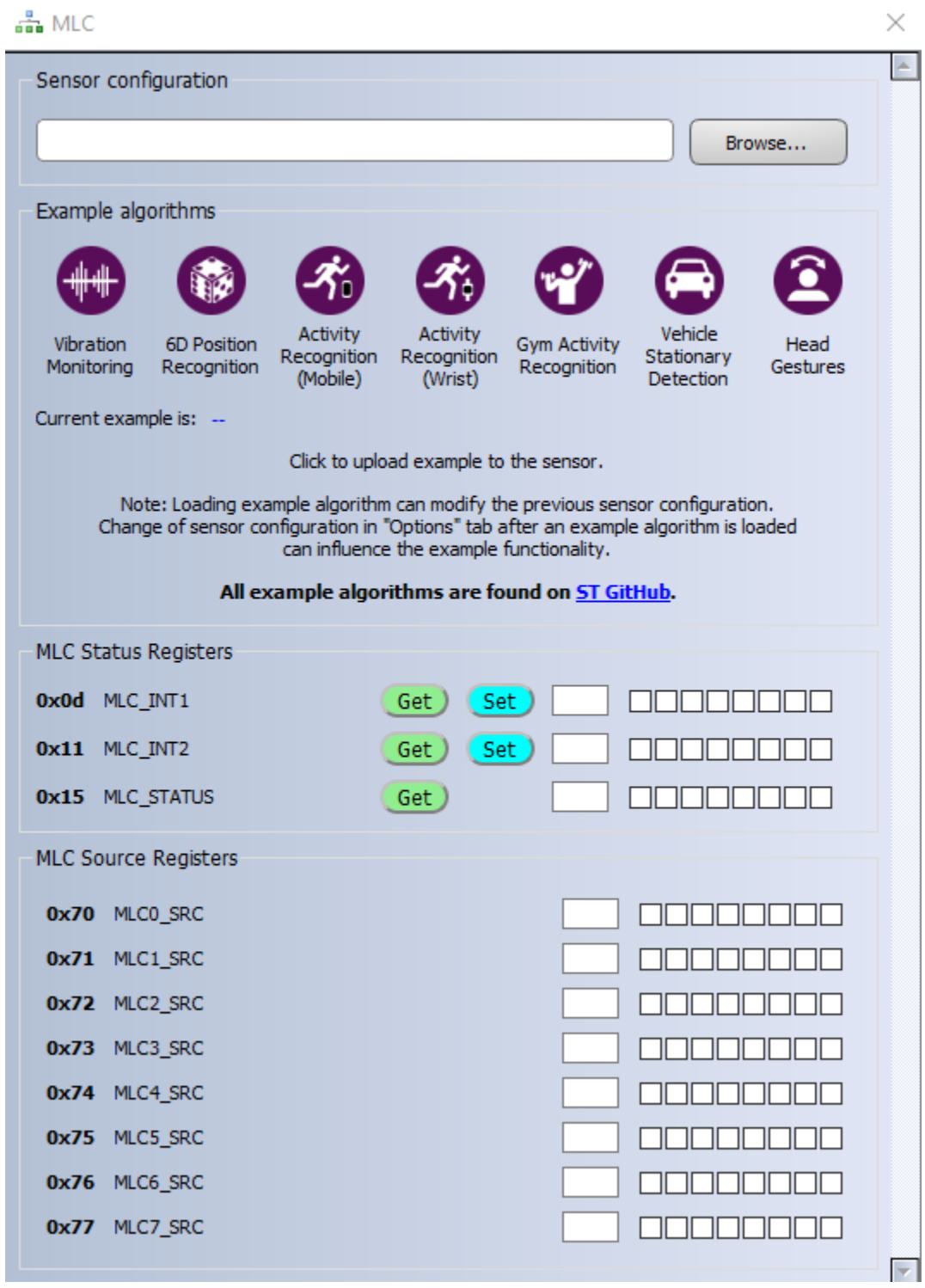
- The sensors list will pop up and show all available sensors. Choose the LSM6DSOX (DIL24).
    - Note that there is also another sensor named LSM6DSO, that is the exaxt same sensor but on the extender IKS01A3 and does not contain the Machine Learning Core.



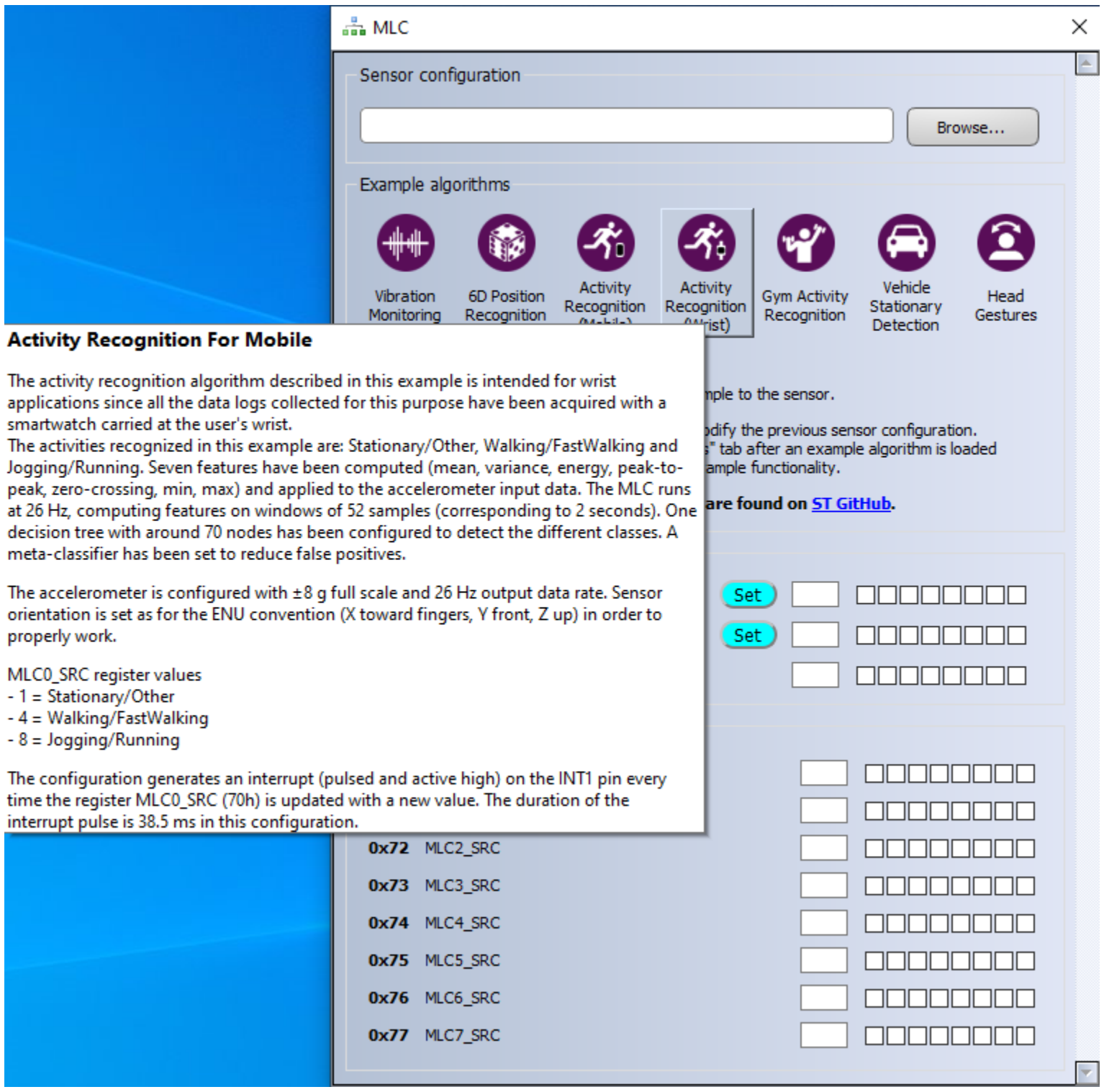
- After `Apply` has been selected, the following window will pop up. It shows the sensors of the IKS01A3 and the MKI197V1 along with their locations.
    - On the left we can see the option for the visualization of the available sensors.



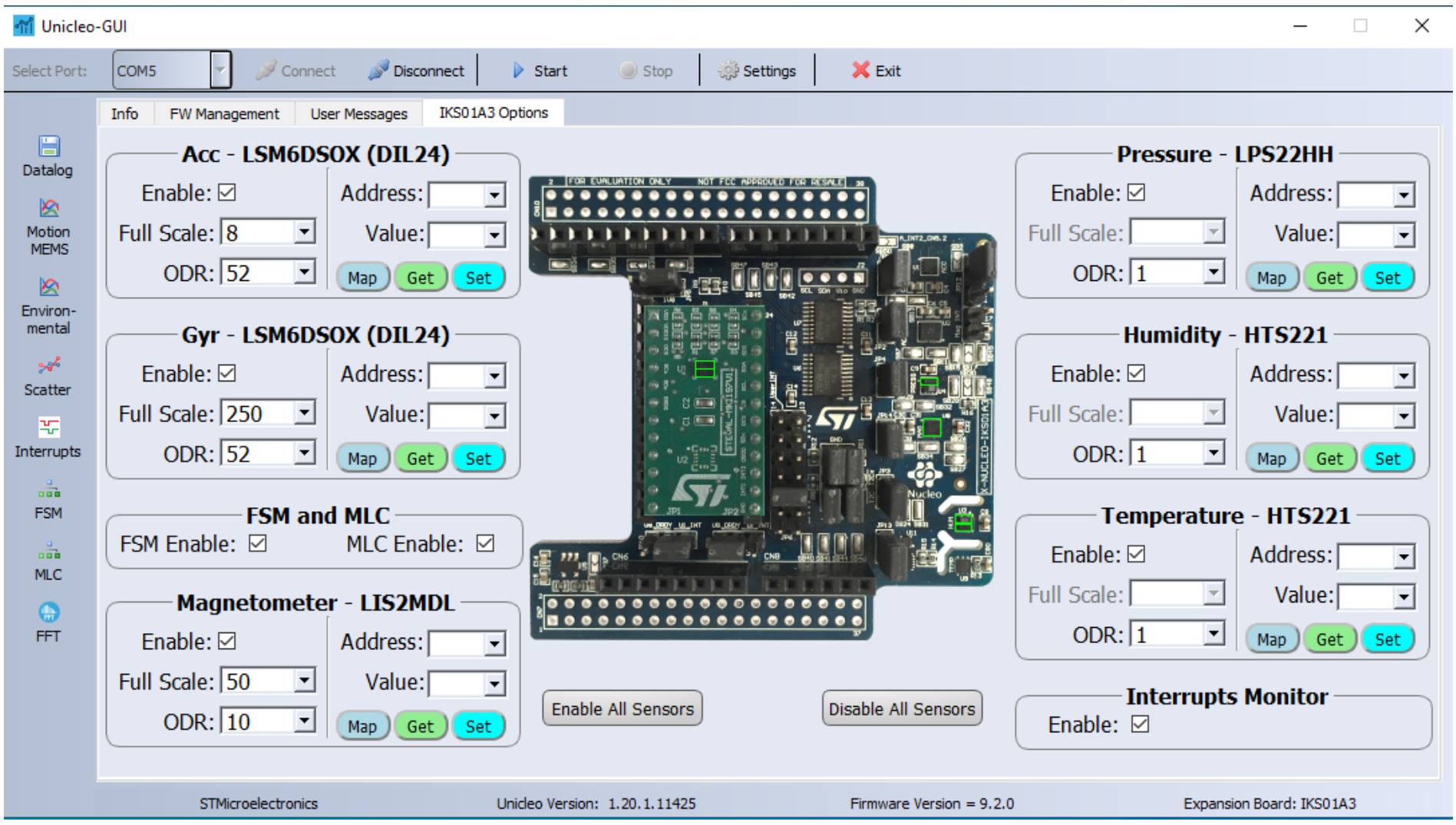
- If `MLC` is chosen on the left, we will be greeted by the following window. On the first block, named `Sensor Configuration`, a custom .ucf dataset can be loaded. Below that are some `Example algorithms` that if chosen they will be loaded and according to the motion of the sensor, a different value will be shown on the `MLC Source Registers` bellow.
    - The values will be displayed on the blocks in the right of `MLC0_SRC`.



- For example, here are the specifics of the `Activity Recognition (Wrist)` algorithm.
    - According to the documentation
        - 1 = Stationary/Other
        - 4 = Walking/FastWalking
        - 8 = Jogging/Running

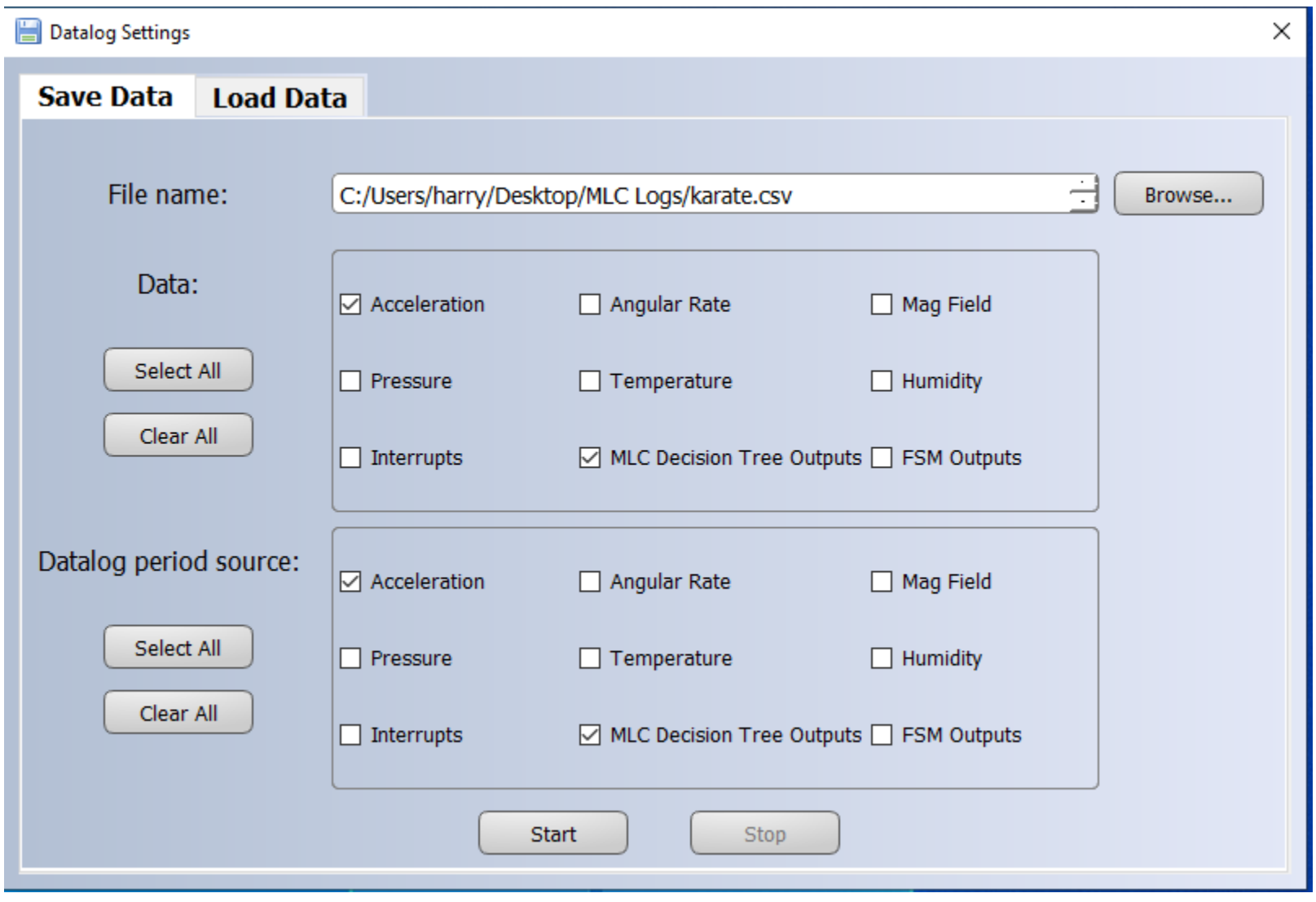


- Before we can select the algorithm we need to select `Start` on the main window. That way we initialize the sensors to begin monitoring.

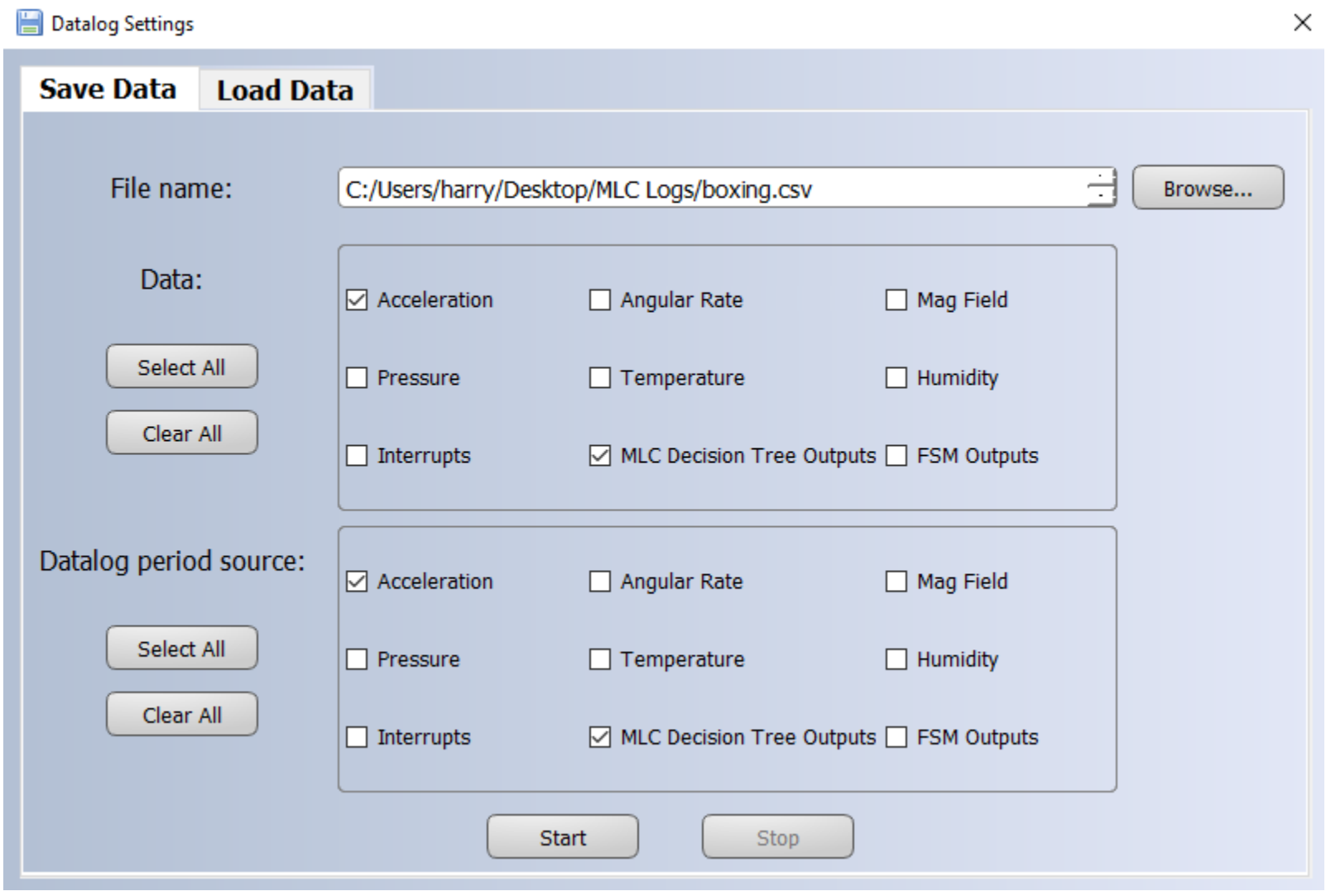


- Now, instead of using one of the example algorithms, I will create my own `.ucf` file. To do that, I need to log data based on each action that I want to add to the Decision Tree. The log will be created in Unicleo (in `.txt/.csv` format) and then I'll use Unico to create the `.ucf` file. Then I'll load the `.ucf` file to Weka to create the Decision Tree and load it back to Unicleo.

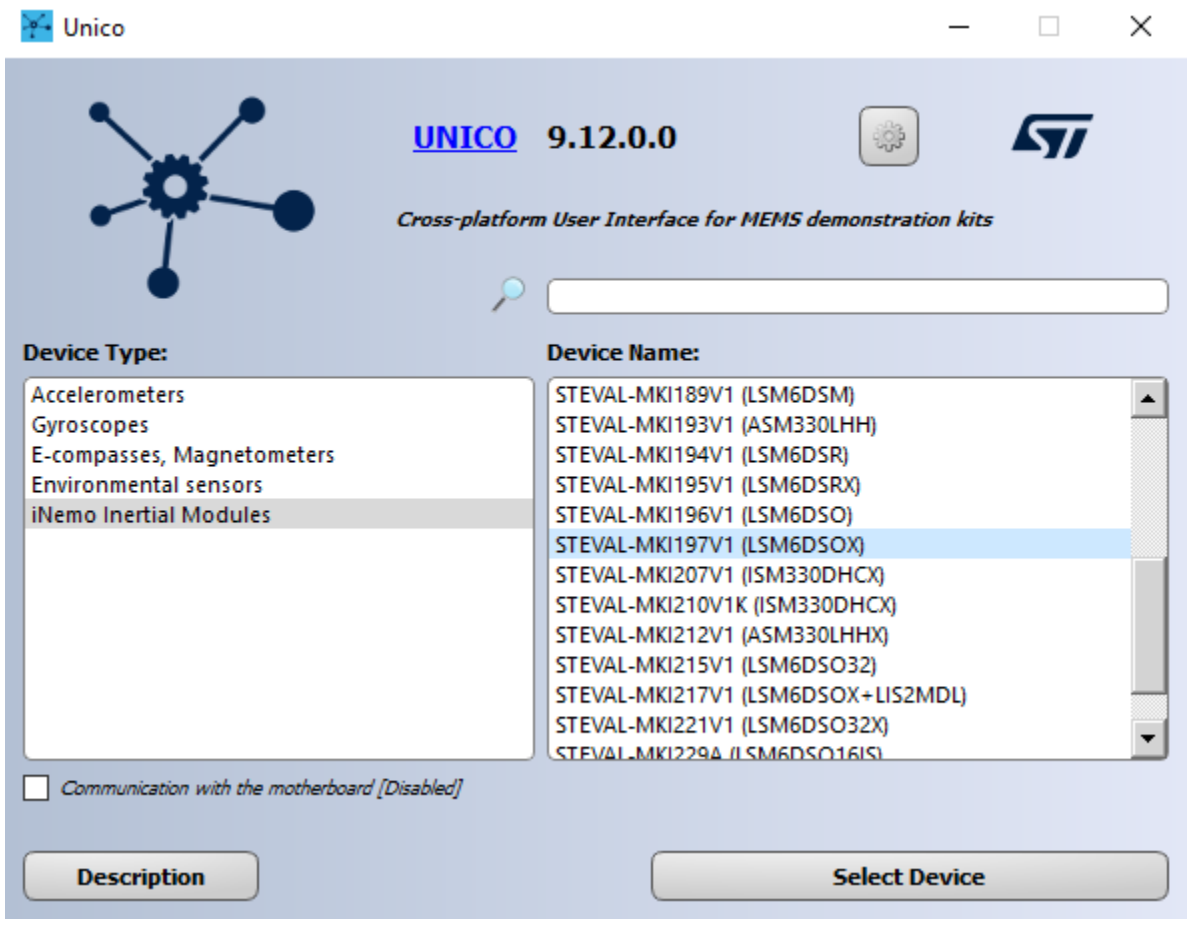
- In order to log the data, the `Datalog` option needs to be chosen on the left on the main window. Next choose the sensors that you want to log into a file from both `Data` and `Datalog period source` and set a file for said activity. Here I will monitor some actions for 1 minute and save them in a file `karate.csv`.



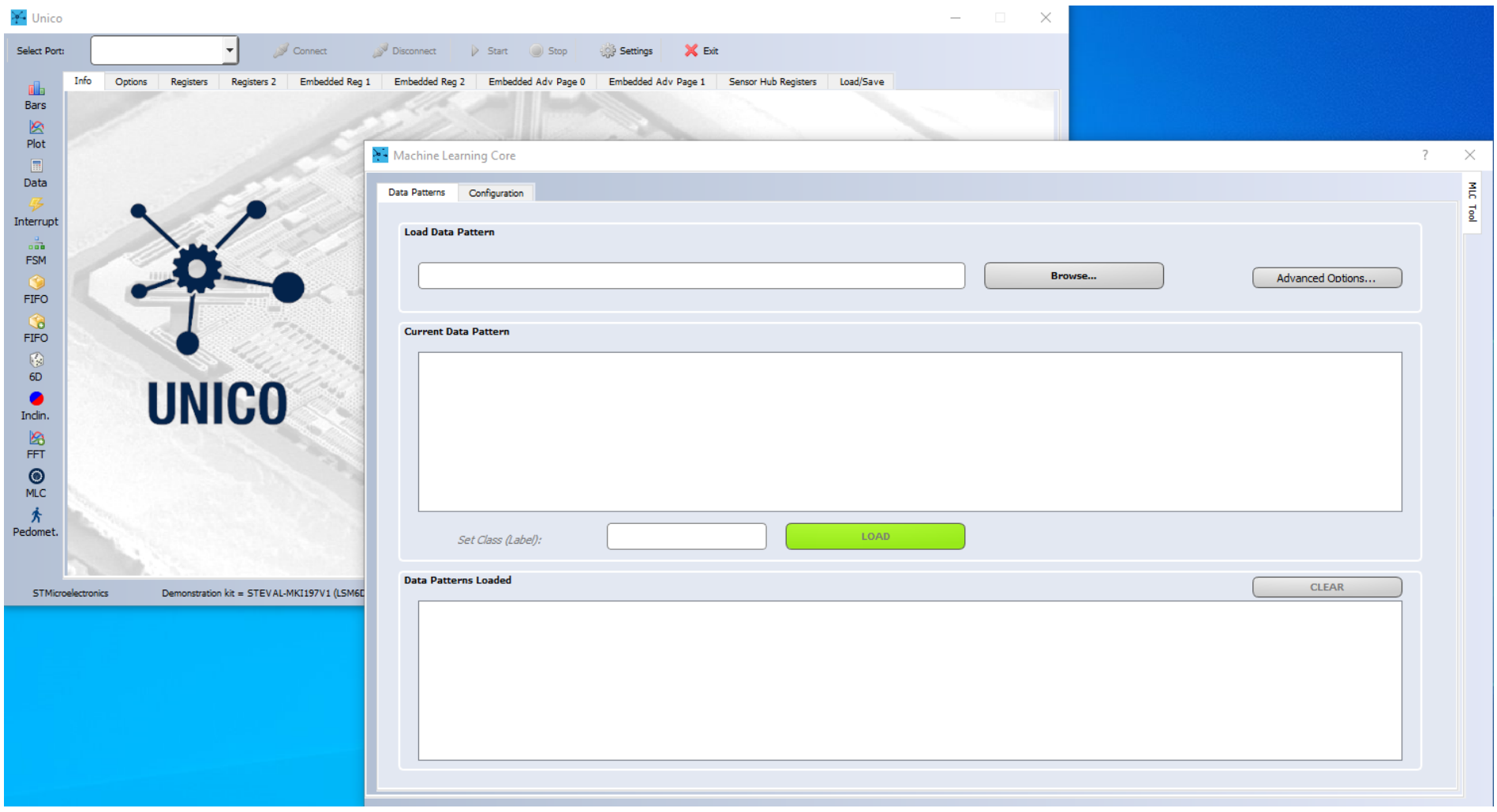
- I will follow the same steps for the dataset `boxing.csv`.



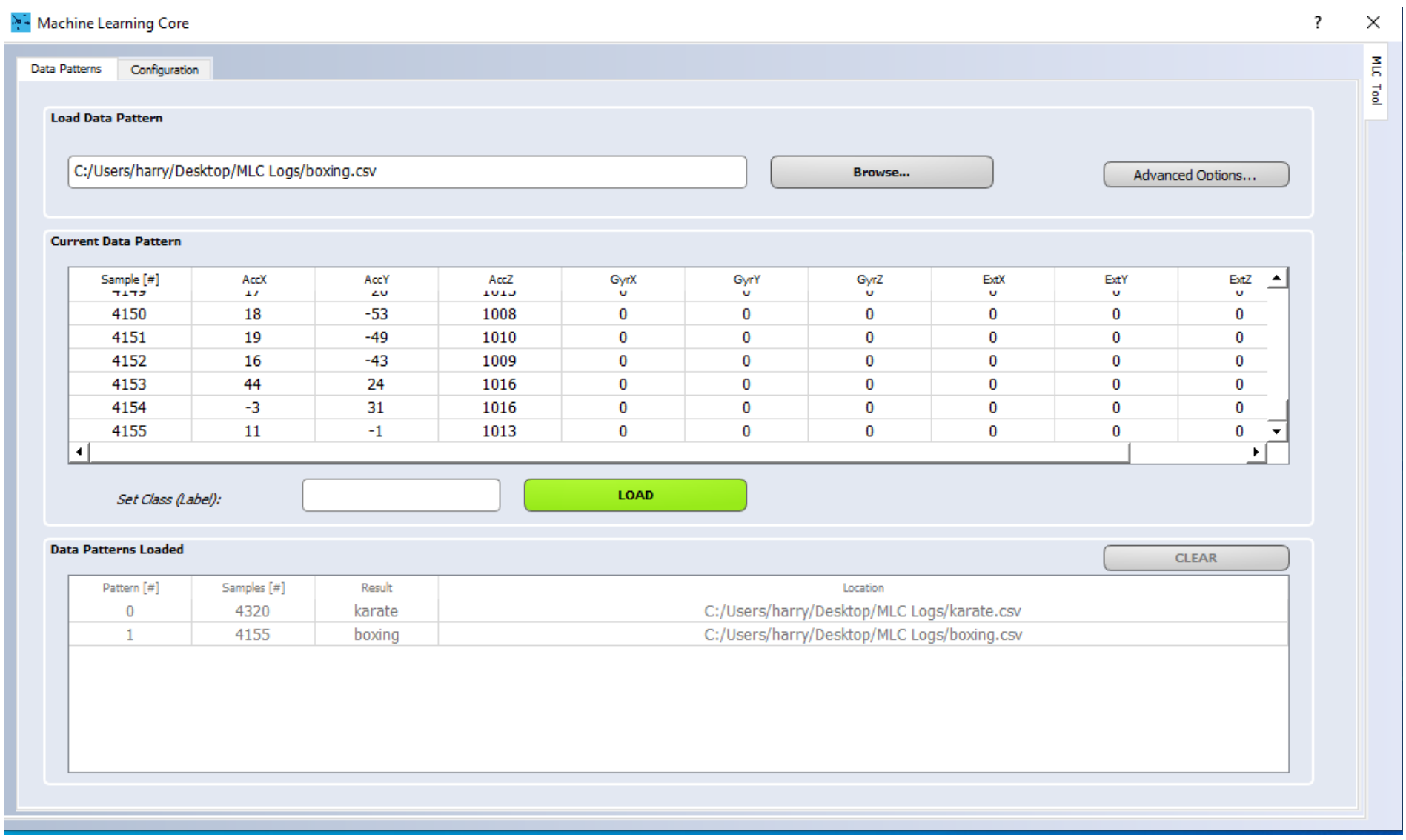
- Now *Unico* needs to be used in order to create a file that can be read by Weka.
    - Since Unico cannot be used with the shield IKS01A3, I will have to use it in offline mode. That means that it will not connect with my board, but I will be able to load my datasets in order to extract an `.arff` file.
    - Select in the `iNemo Inertial Modules` the `STEVAL-MKI197V1 (LSM6DSOX)` sensor and diselect the `Communication with the motherboard` option. Then click on `Select Device`.



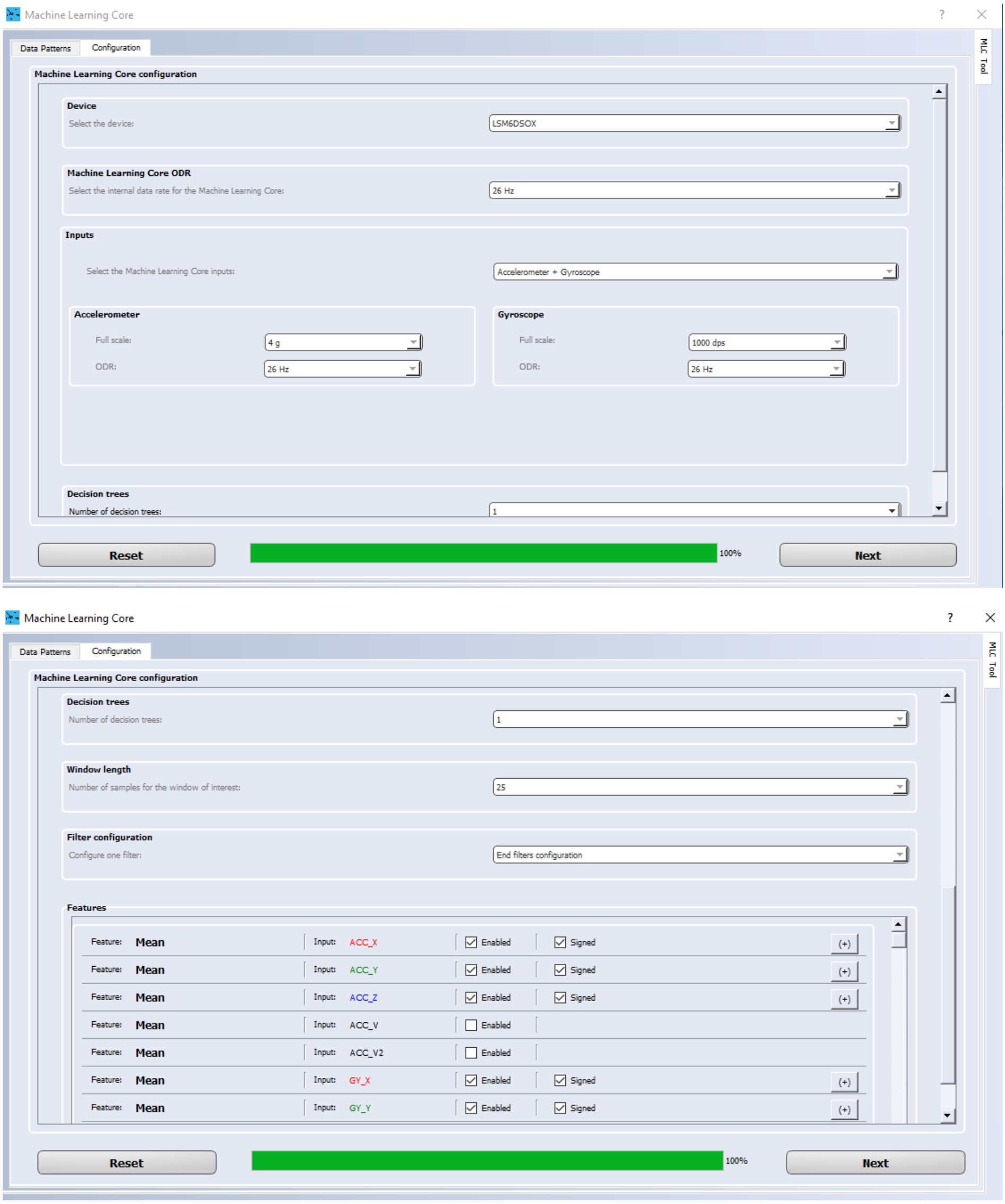
- After the main window shows up, select `MLC` on the left and the Machine Learning Core window will open.

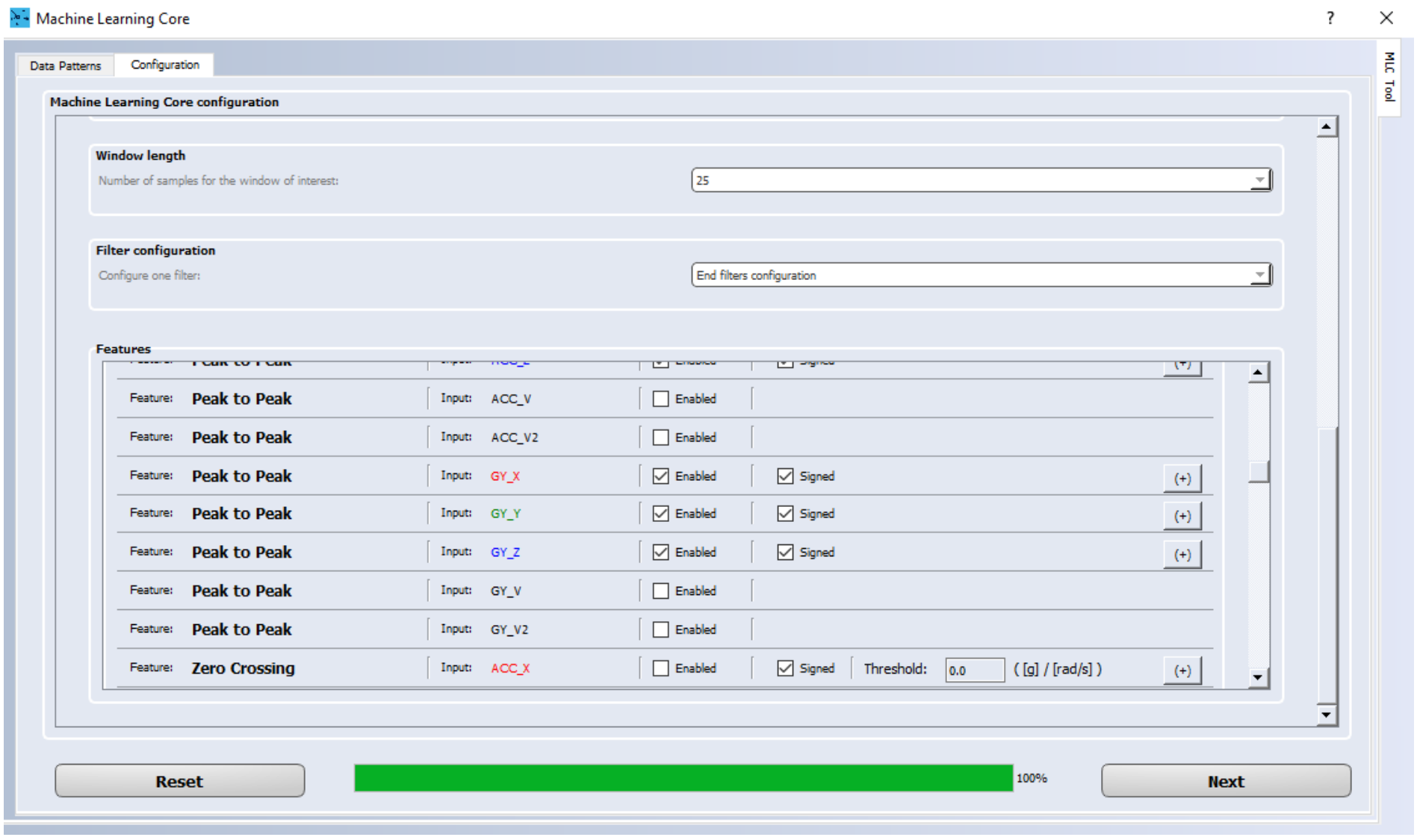


- Now we load each one of the datasets and set the `Class (Label)` as `boxing` for the boxing dataset and `karate` for the karate dataset.

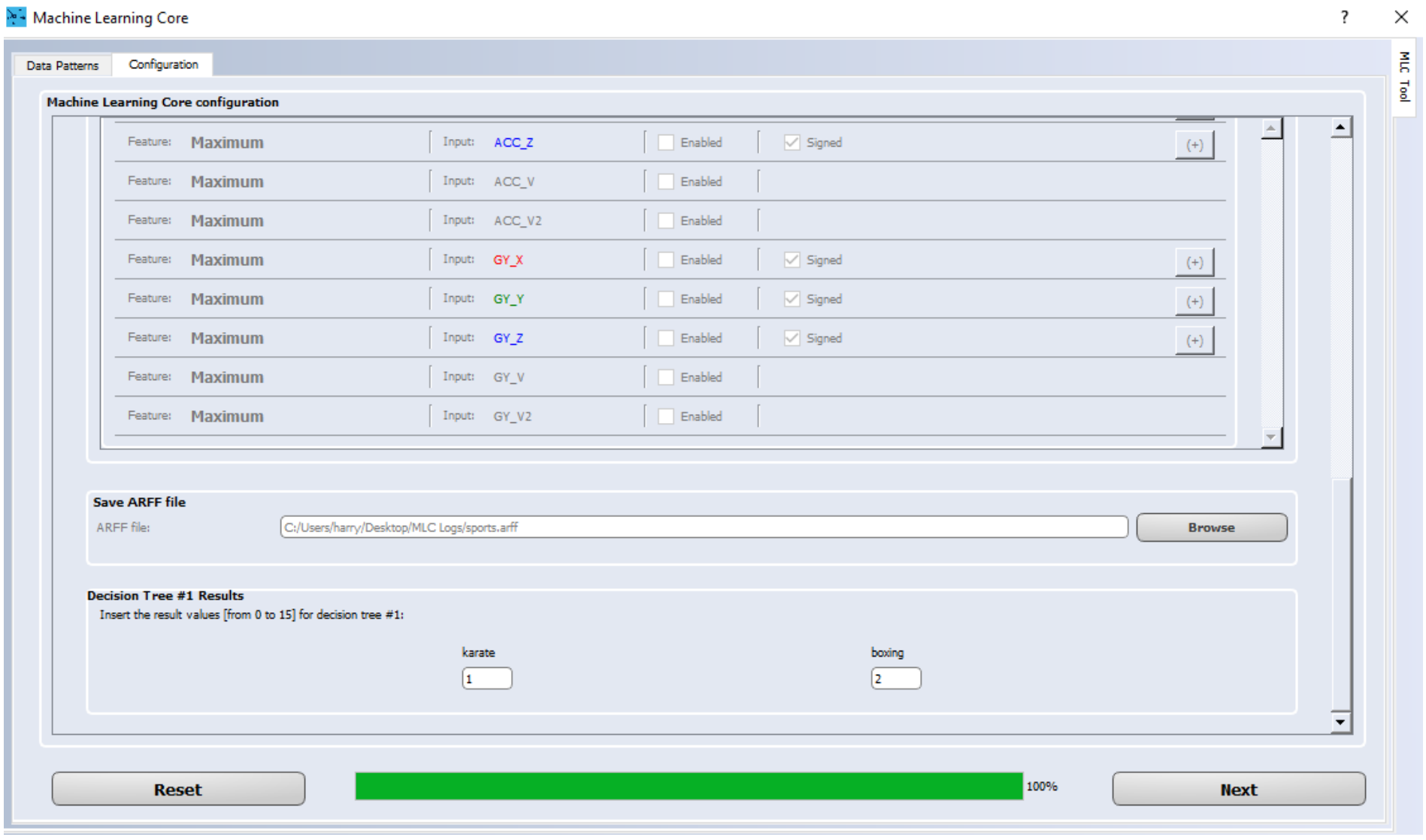


- Then we move to the `Configuration` Tab. Here are the options that I gave to my Decision Tree.

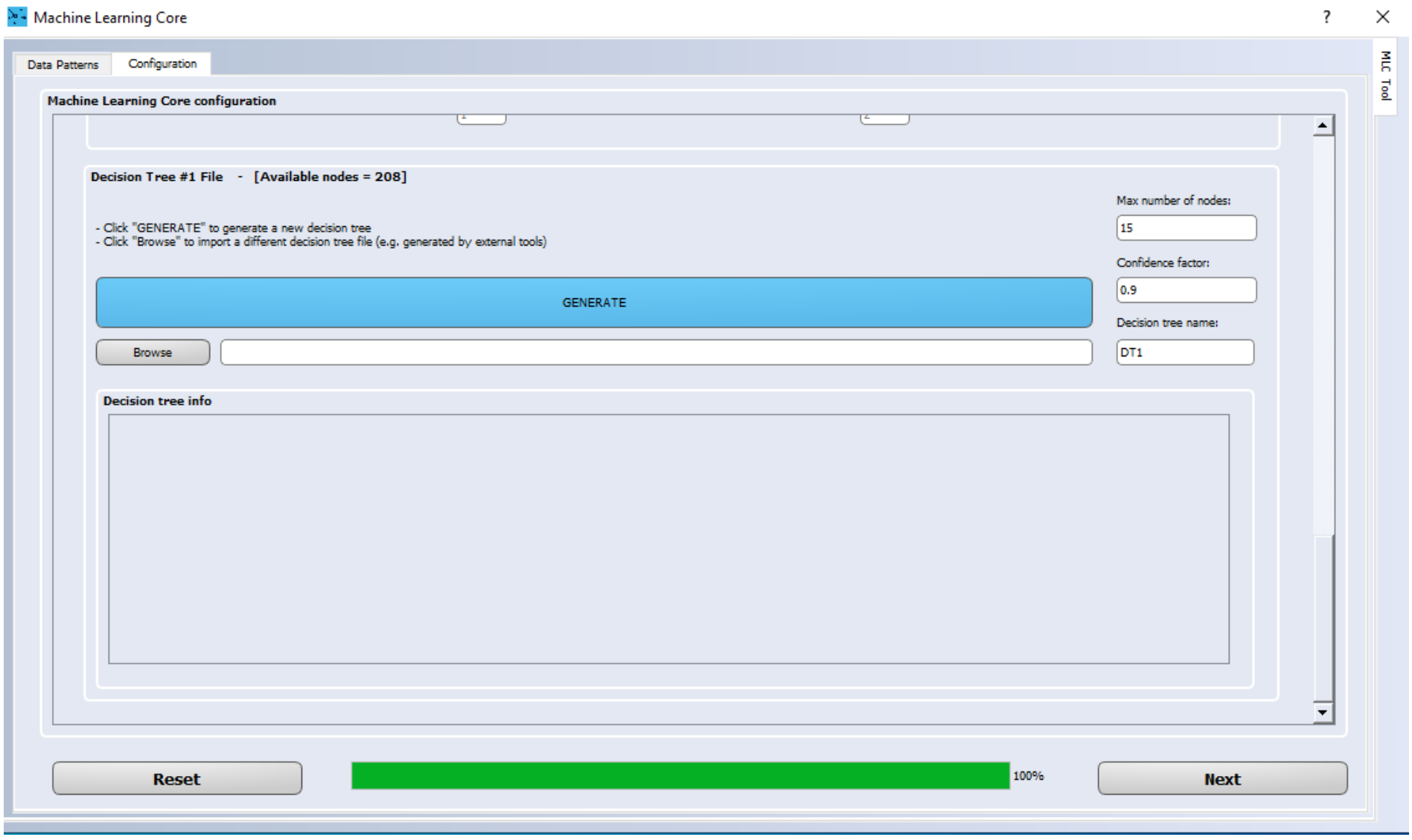

Machine Learning Core
Data Patterns
Configuration
MLC Tool
Machine Learning Core configuration
Device
Select the device:
LSM6DSOX
Machine Learning Core ODR
Select the internal data rate for the Machine Learning Core:
26 Hz
Inputs
Select the Machine Learning Core inputs:
Accelerometer + Gyroscope
Accelerometer
Full scale:
4 g
ODR:
26 Hz
Gyroscope
Full scale:
1000 dps
ODR:
26 Hz
Decision trees
Number of decision trees:
1
Reset
100%
Next
Machine Learning Core
Data Patterns
Configuration
MLC Tool
Machine Learning Core configuration
Decision trees
Number of decision trees:
1
Window length
Number of samples for the window of interest:
25
Filter configuration
Configure one filter:
End filters configuration
Features
Feature: Mean Input: ACC_X Enabled Signed (+)
Feature: Mean Input: ACC_Y Enabled Signed (+)
Feature: Mean Input: ACC_Z Enabled Signed (+)
Feature: Mean Input: ACC_V Enabled
Feature: Mean Input: ACC_V2 Enabled
Feature: Mean Input: GY_X Enabled Signed (+)
Feature: Mean Input: GY_Y Enabled Signed (+)
Reset
100%
Next



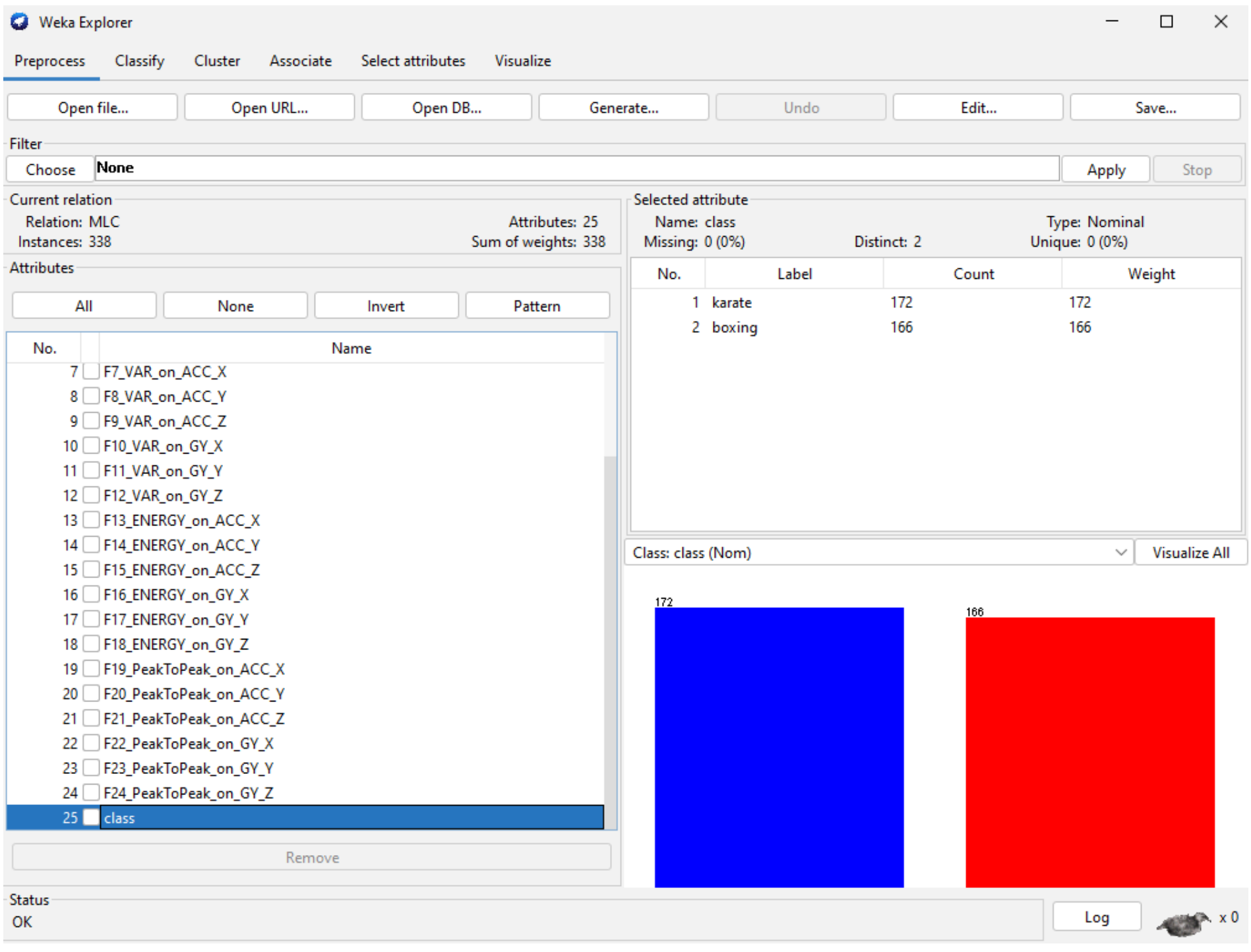
- I chose all the *Signed* options `ACC_X`, `ACC_Y`, `ACC_Z`, `GY_X`, `GY_Y`, `GY_Z` for the `Mean`, `Variance`, `Energy` and `Peak to Peak` features.



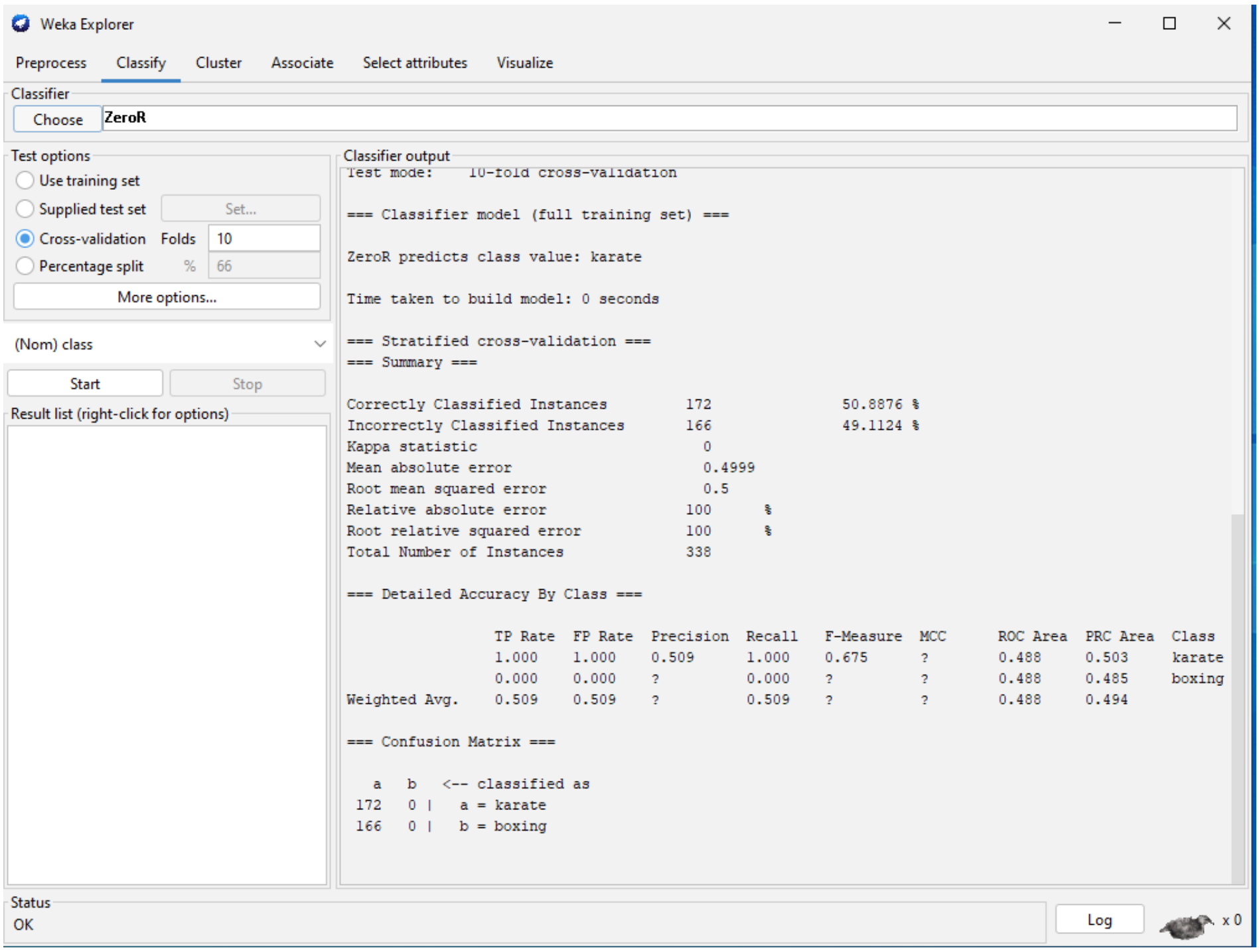
- Here I name a file to save as `.arff` and choose the output that I want to see in the Decision Tree later.
  - I set `1` for karate and `2` for boxing.

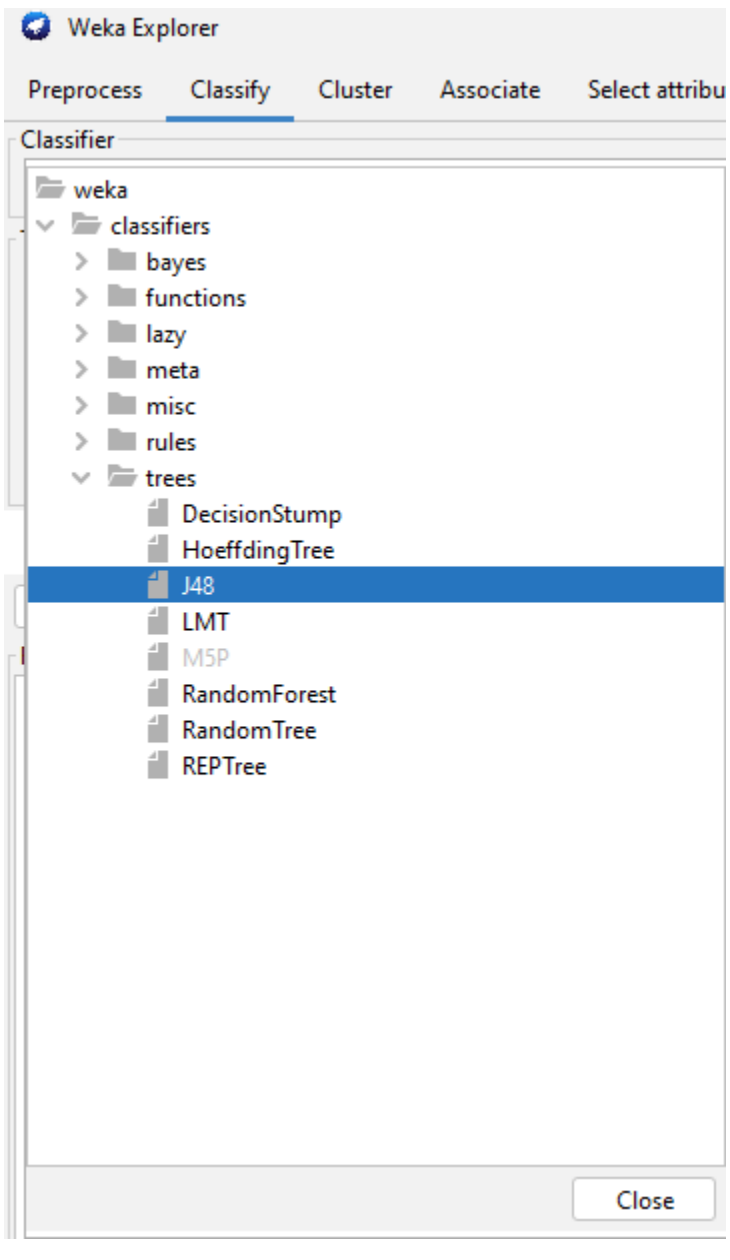


- Open Weka Explorer and load the `sports.arff` file we created above:

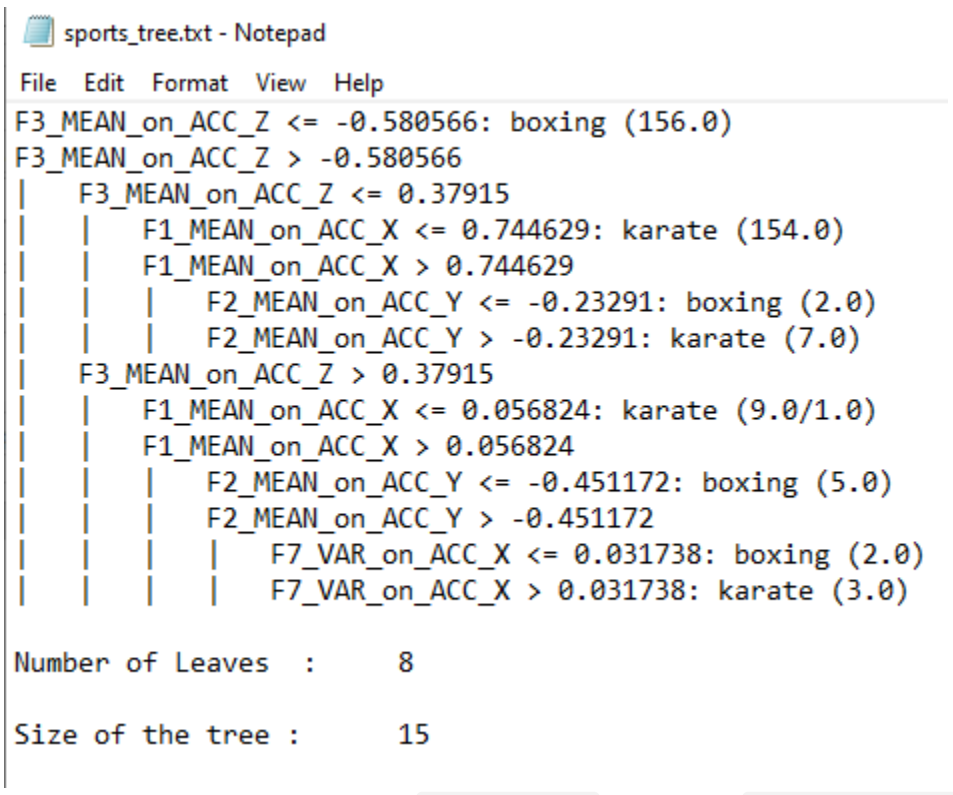


- Select the `Clasify Tab` and then `Choose` on the `Classifier` Block and select the `J48` in the tree section. I let the `Cross-Validation Folds` on 10 as default since it gives a ~98% result.

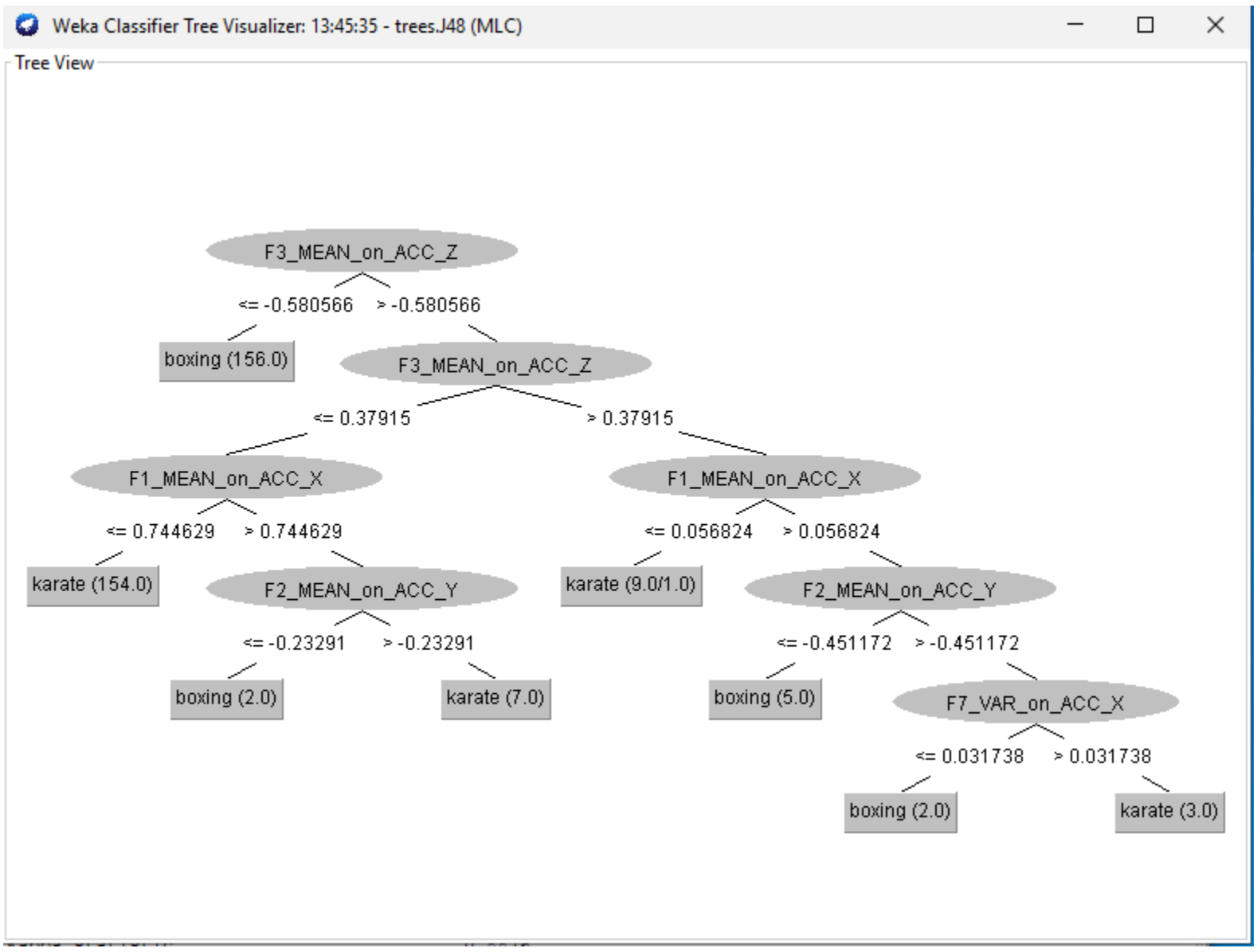

Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
ZeroR
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
(Nom) class
Start
Stop
Result list (right-click for options)
Classifier output
Test mode: 10-fold cross-validation
=== Classifier model (full training set) ===
ZeroR predicts class value: karate
Time taken to build model: 0 seconds
=== Stratified cross-validation ===
=== Summary ===
Correctly Classified Instances 172 50.8876 %
Incorrectly Classified Instances 166 49.1124 %
Kappa statistic 0
Mean absolute error 0.4999
Root mean squared error 0.5
Relative absolute error 100 %
Root relative squared error 100 %
Total Number of Instances 338
=== Detailed Accuracy By Class ===
TP Rate FP Rate Precision Recall F-Measure MCC ROC Area PRC Area Class
1.000 1.000 0.509 1.000 0.675 ? 0.488 0.503 karate
0.000 0.000 ? 0.000 ? ? 0.488 0.485 boxing
Weighted Avg. 0.509 0.509 ? 0.509 ? ? 0.488 0.494
=== Confusion Matrix ===
a b <-- classified as
172 0 | a = karate
166 0 | b = boxing
Status
OK
Log
x 0


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attribu
Classifier
weka
classifiers
bayes
functions
lazy
meta
misc
rules
trees
DecisionStump
HoeffdingTree
J48
LMT
M5P
RandomForest
RandomTree
REPTree
Close

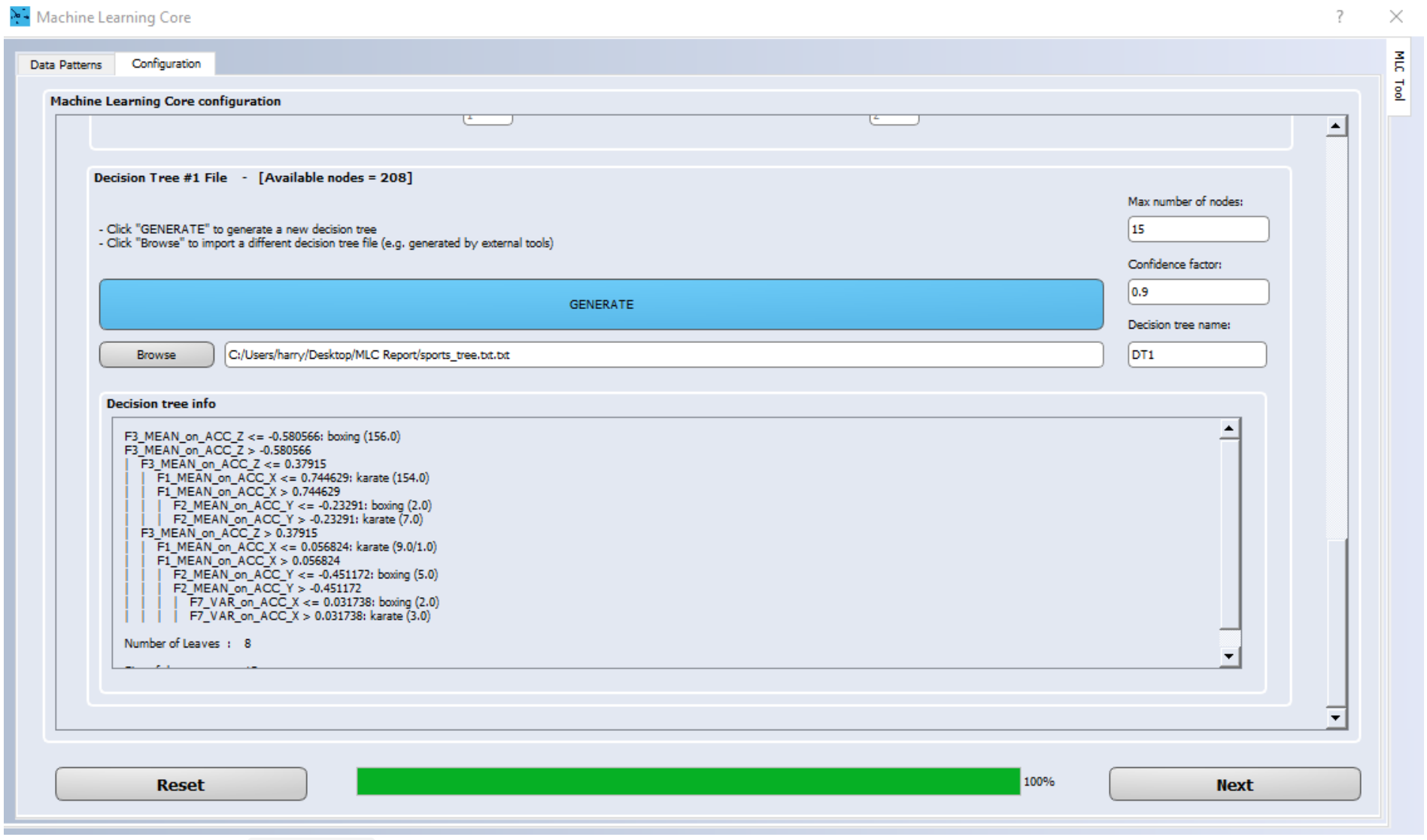
- The Decision Tree has be generated. But in order to load it into Unico to create our `.ucf` file we need to copy the selected text as in the image below (The tree itself) and paste it into a `.txt` file.



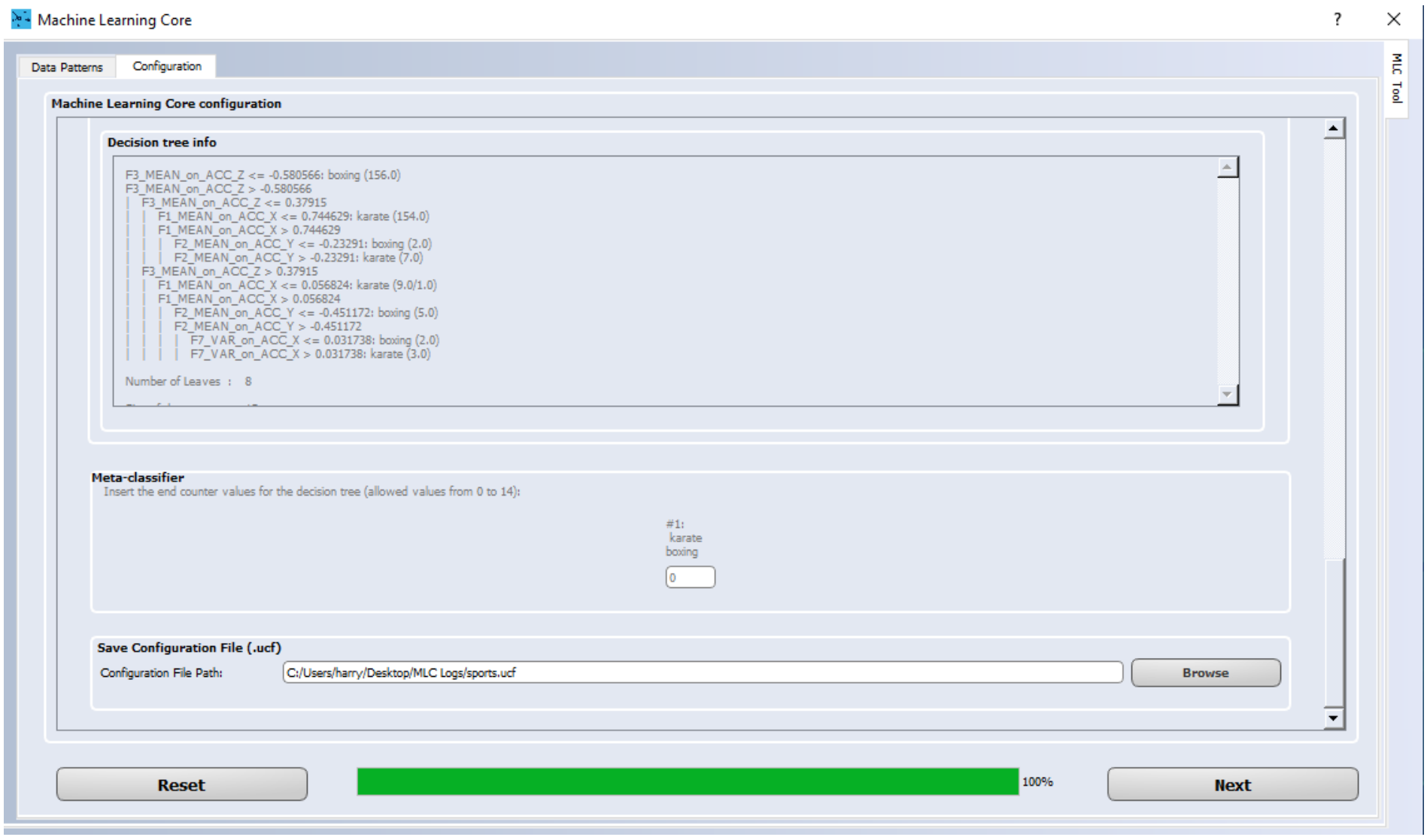
- And if we want to see the Decision Tree, then right click on the `Result list` and select `Visualize Tree`.



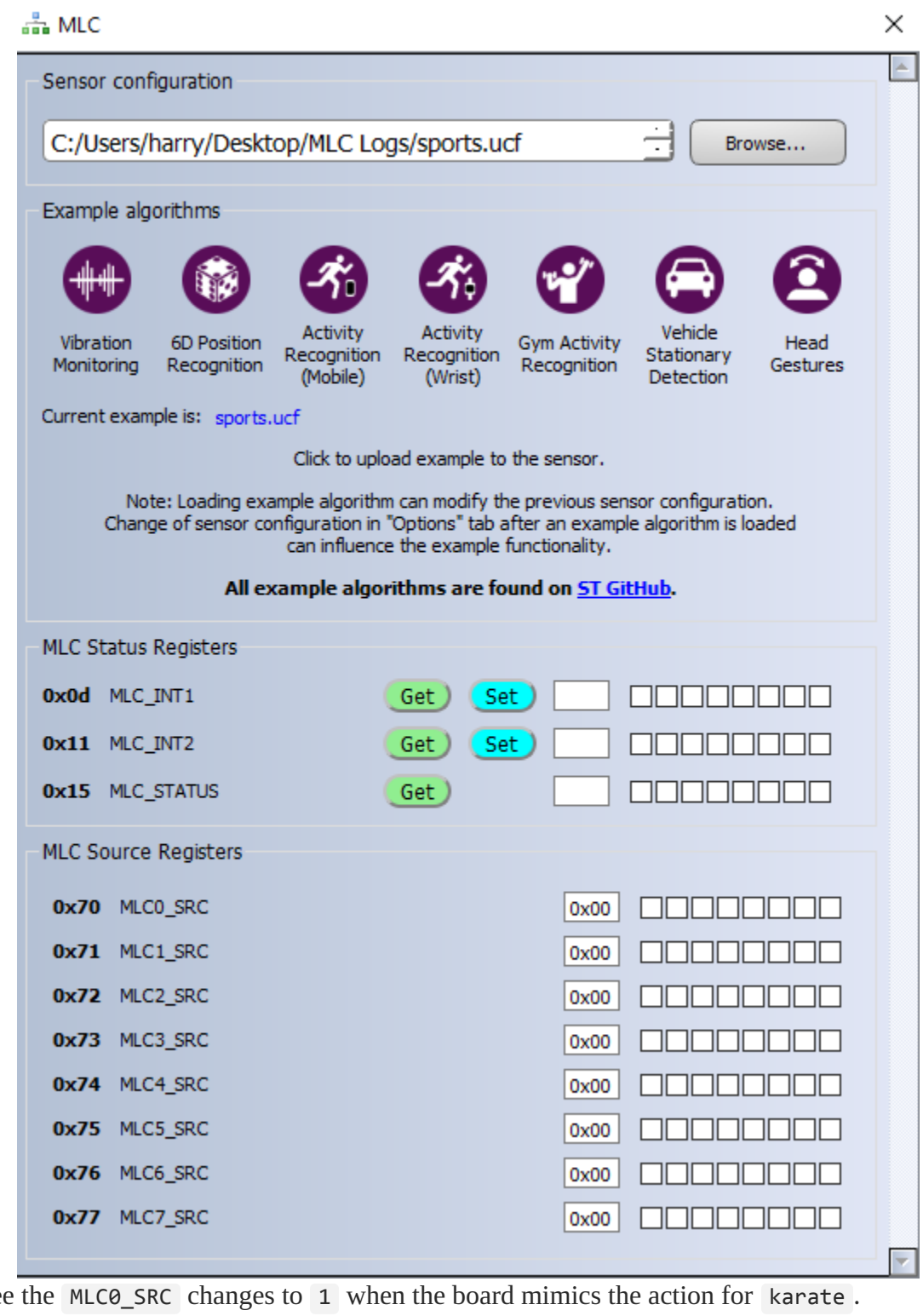
- Now we can load the `.txt` file into the Unico window we left intact a while ago.



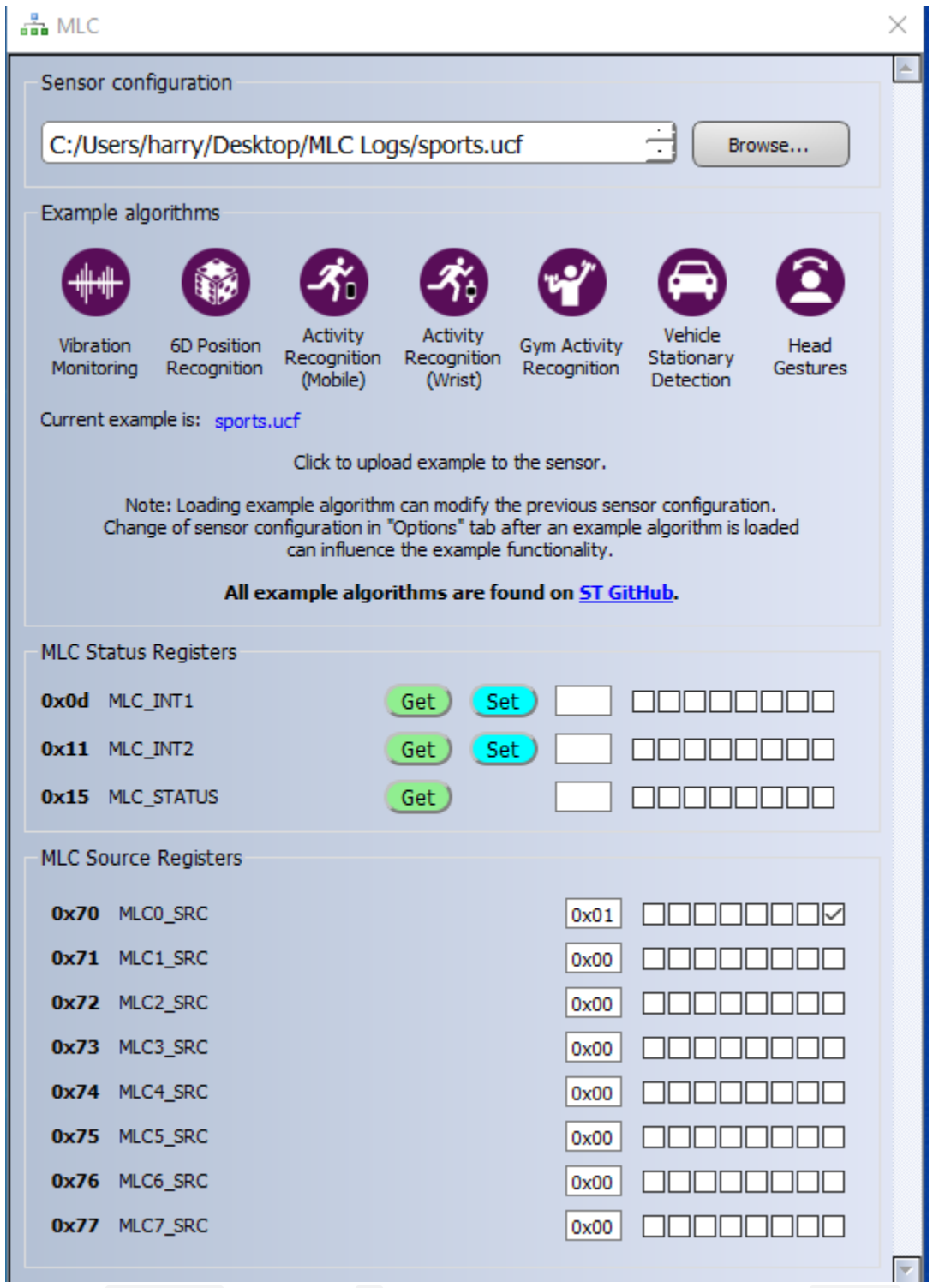
- And save the file as `sports.ucf`.



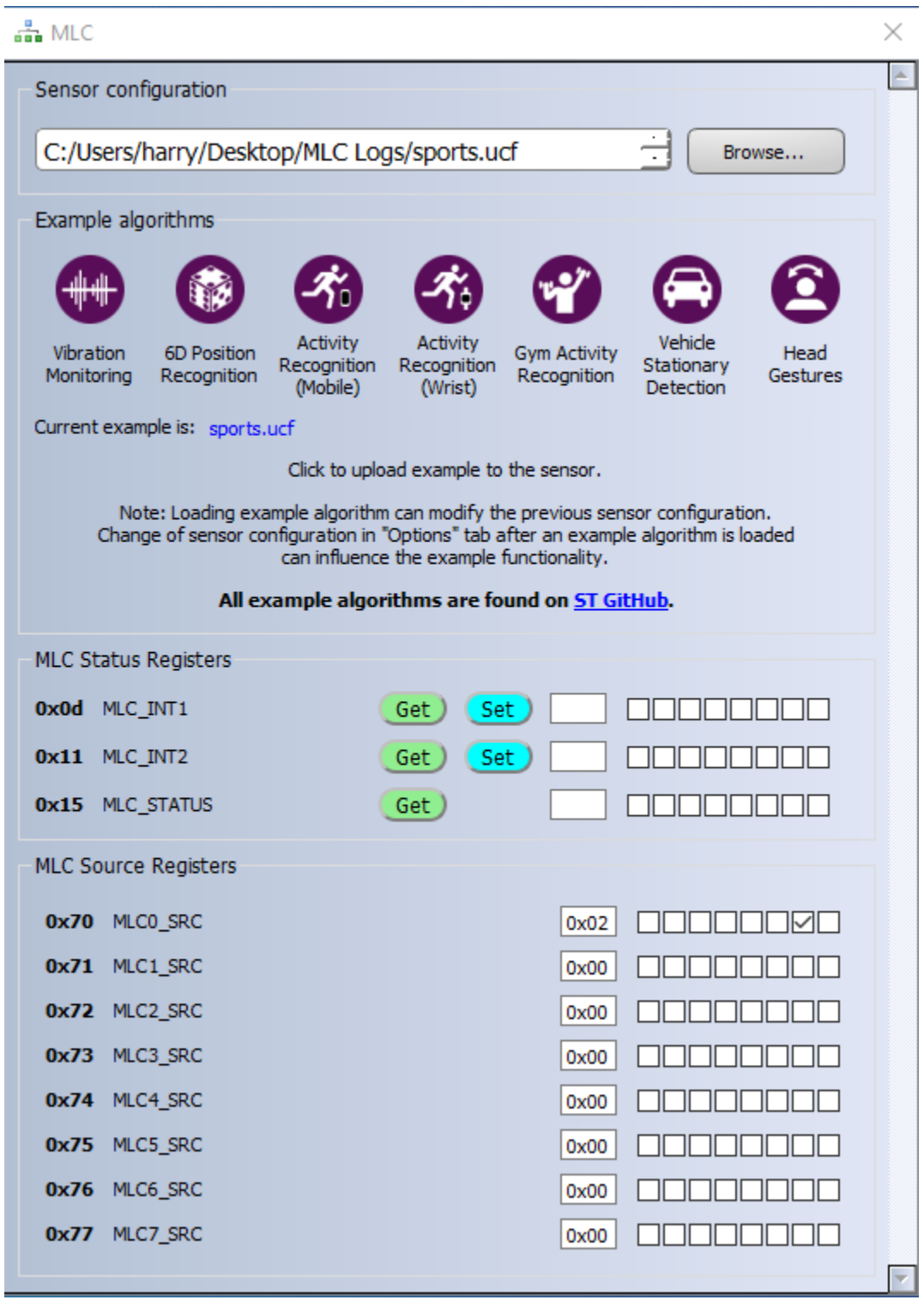
- Now we can go back to Unicleo and load the `sports.ucf` file to the `MLC`.



  - Here as we can see the `MLC0_SRC` changes to `1` when the board mimics the action for `karate`.



- And here as we can see the `MLC0_SRC` changes to `2` when the board mimics the action for `boxing`.



- Here is a video demostration of the LSM6DSOX sensor changing value from `1` to `2` when it recognises the different actions.

  Machine Learning Core LSM6DSOX Demonstration || ISCA Lab

This concludes this demo.

- Some notes on the `Machine Learning Core` and the `Finite State Machine` options on Unicleo/Unico.
  - `Finite State Machine` gives only a True or False result, from my understanding. This means that it can only detect whether the sensor is Idle or in a specific Action, it cannot distinguish between different events and actions.
  - Whereas the Machine Learning Core **can** distinguish between different actions performed by the sensor. For example it can tell us when the sensor is Idle or when a specific action (from numerous loaded in the Decision Tree) is performed.