

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**).

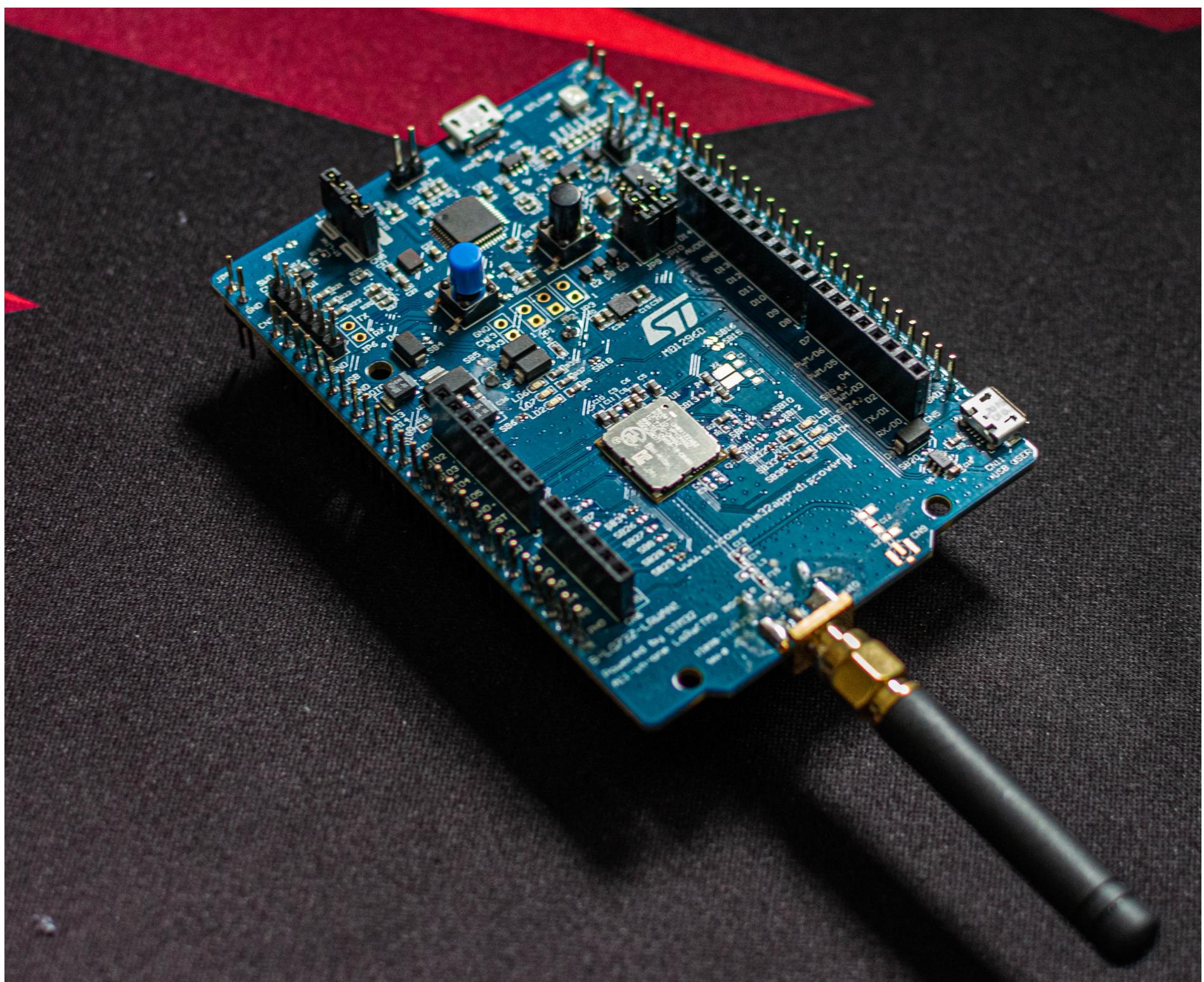
The project linked above needs to be programmed and flashed with the STM32CubeIDE program, since it does not include a CubeMX project.

The **Software** used is:

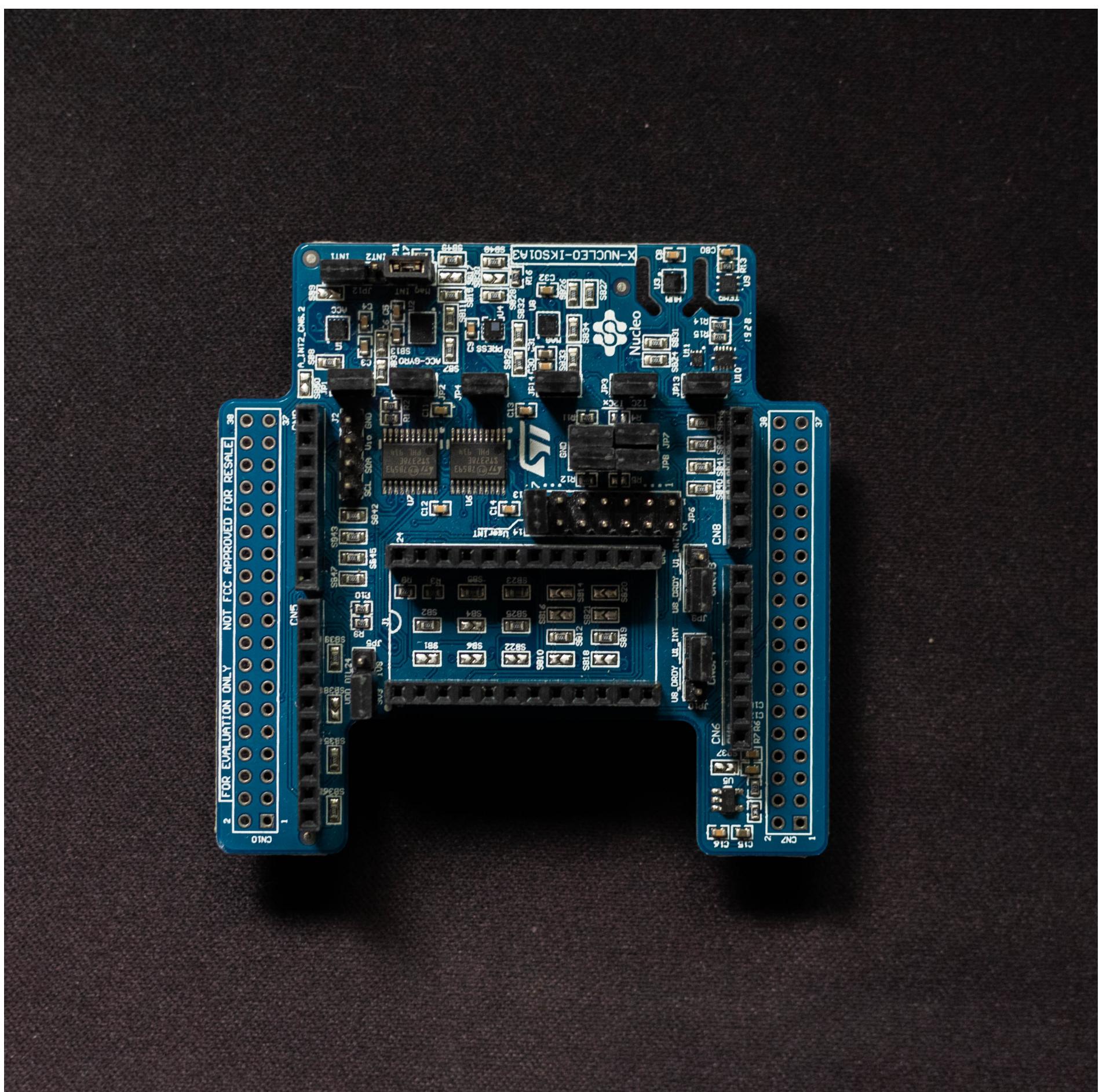
1. **Unicleo-GUI:** Demonstrates the functionality of ST sensors and algorithms
2. **Unico:** Needed to create the .ucf file that will be loaded after the Decision Tree is created.
3. **Weka:** Machine Learning Tool used for Data Mining tasks.

Hardware Combination and Specifications.

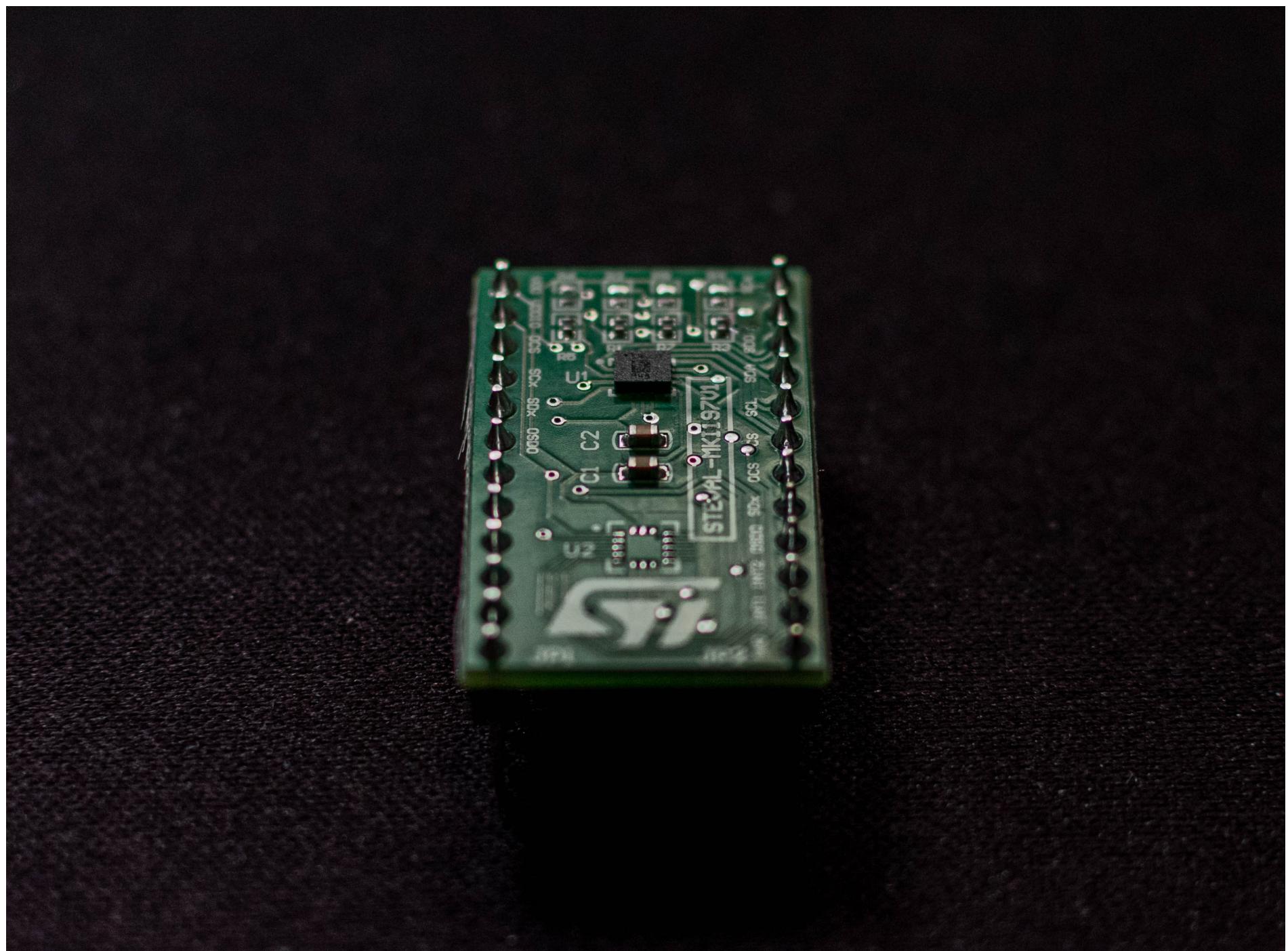
1. **B-L072Z-LRWAN1**



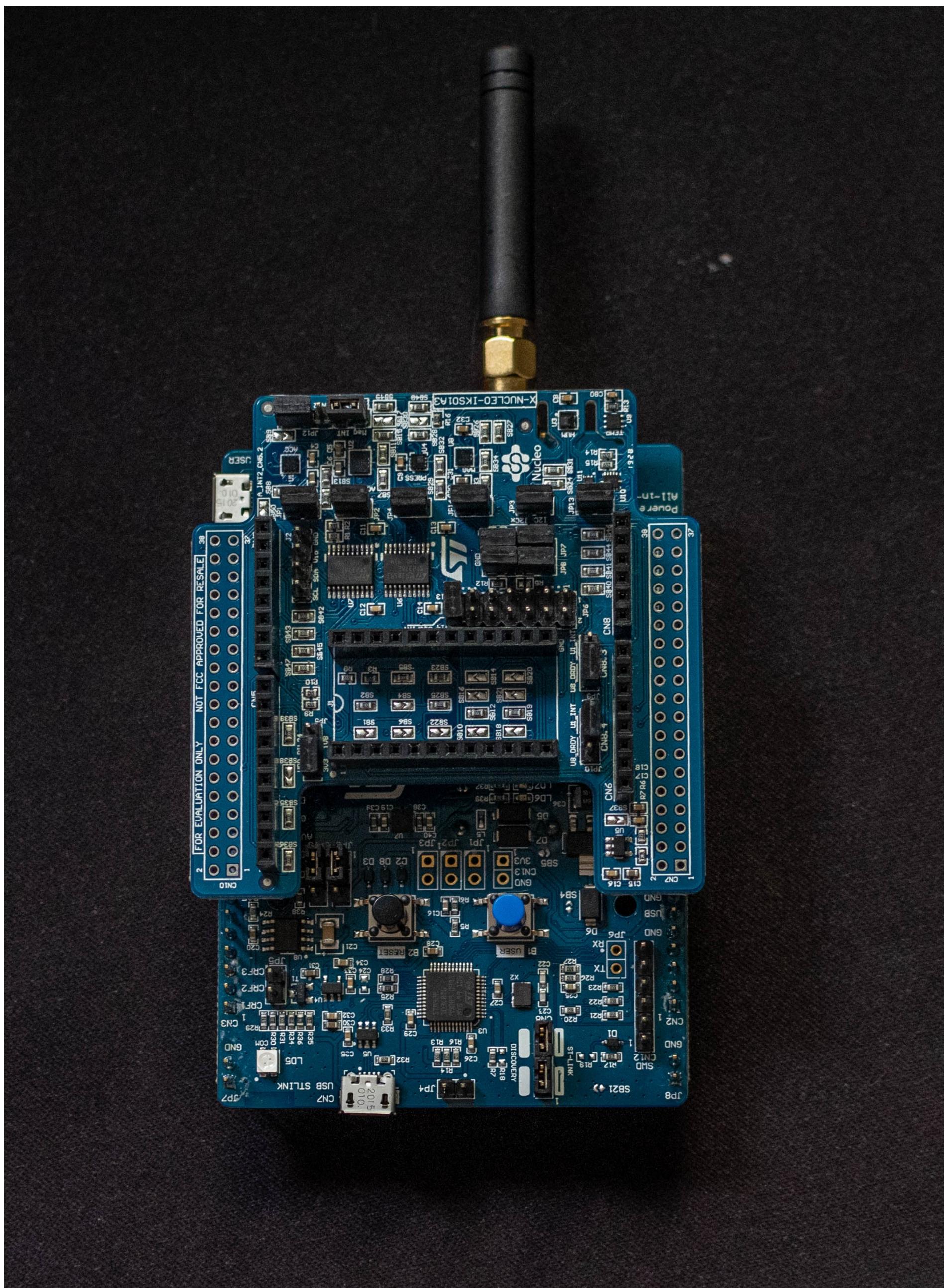
2. X-NUCLEO-IKS01A3



3. MKI197V1 (LSM6DSOX)

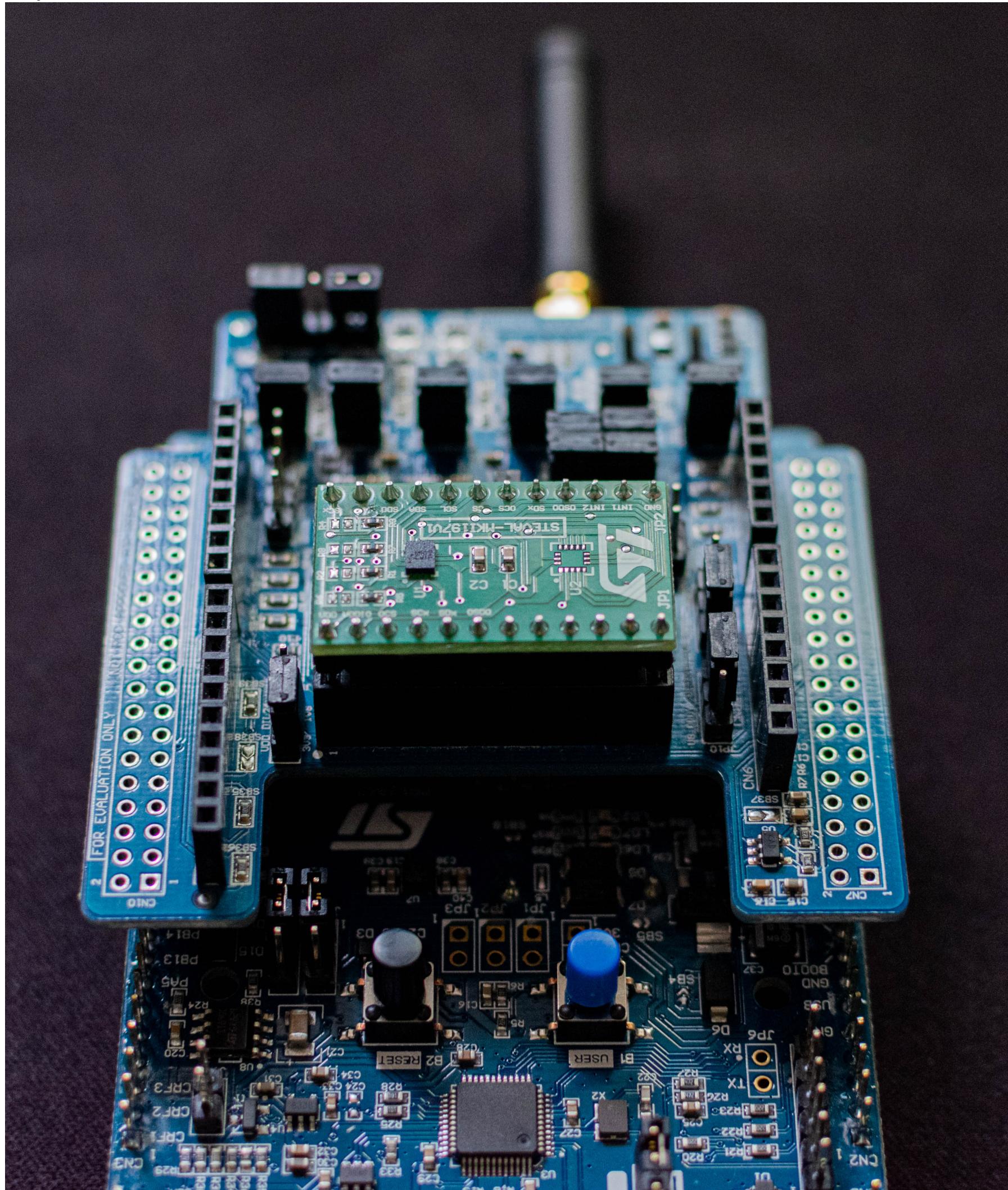


- First we need to combine the X-NUCLEO-IKS01A3 on the Arduino R3 pinouts of B-L072Z-LRWAN1

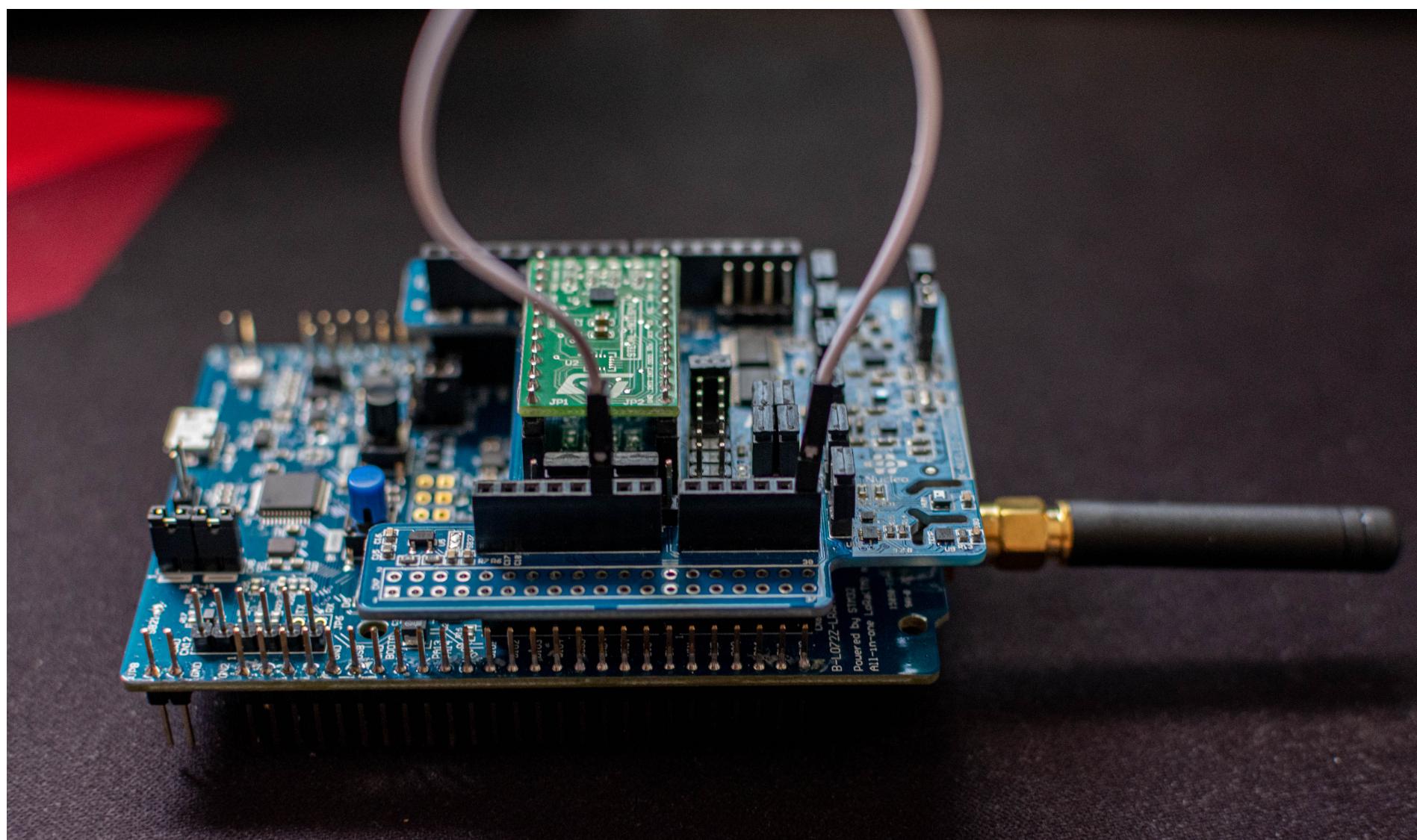


- Next we add the MKI197V1 on the DIL24 Socket of the IKS01A3

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 I₂C mode because of a level shifter on the IKS01A3 that keeps the INT1 of the LSM6DSOX high, this results to I₂C 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.

The screenshot shows the STM32CubeIDE interface. The Project Explorer on the left lists several projects and files, including 'B-L475-IOT01A2_I2C_Sniffer', 'DataLogExtended (in STM32CubeIDE)', 'STM32L073RZTx_FLASH.ld', and various source and header files. The main window displays the 'STM32L073RZTx_FLASH.ld' linker script file. The code in the script is as follows:

```

1 /*
2 ****
3 ** File      : LinkerScript.ld
4 ** Author    : Auto-generated by STM32CubeIDE
5 **
6 ** Abstract  : Linker script for STM32L073RZTx Device from stm32l0 series
7 **              192Kbytes ROM
8 **              20Kbytes RAM
9 **
10 **          Set heap size, stack size and stack location according
11 **          to application requirements.
12 **          Set memory bank area and size if external memory is used.
13 **          Target    : STMicroelectronics STM32
14 **          Distribution: The file is distributed as is without any warranty
15 **                      of any kind.
16 **          Redistribution and use in source and binary forms, with or without modification,
17 **          are permitted provided that the following conditions are met:
18 **          1. Redistributions of source code must retain the above copyright notice,
19 **             this list of conditions and the following disclaimer.
20 **          2. Redistributions in binary form must reproduce the above copyright notice,
21 **             this list of conditions and the following disclaimer in the documentation
22 **             and/or other materials provided with the distribution.
23 **          3. Neither the name of STMicroelectronics nor the names of its contributors
24 **             may be used to endorse or promote products derived from this software
25 **             without specific prior written permission.
26 */
27 ****
28 ** @attention
29 ** <h2><center>&copy; COPYRIGHT(c) 2020 STMicroelectronics</center></h2>
30 ** Redistribution and use in source and binary forms, with or without modification,
31 ** are permitted provided that the following conditions are met:
32 ** 1. Redistributions of source code must retain the above copyright notice,
33 **     this list of conditions and the following disclaimer.
34 ** 2. Redistributions in binary form must reproduce the above copyright notice,
35 **     this list of conditions and the following disclaimer in the documentation
36 **     and/or other materials provided with the distribution.
37 ** 3. Neither the name of STMicroelectronics nor the names of its contributors
38 **     may be used to endorse or promote products derived from this software
39 **     without specific prior written permission.
40 */
41 STM32L073RZTx_FLASH.ld

```

The bottom console window shows the output of the build process:

```

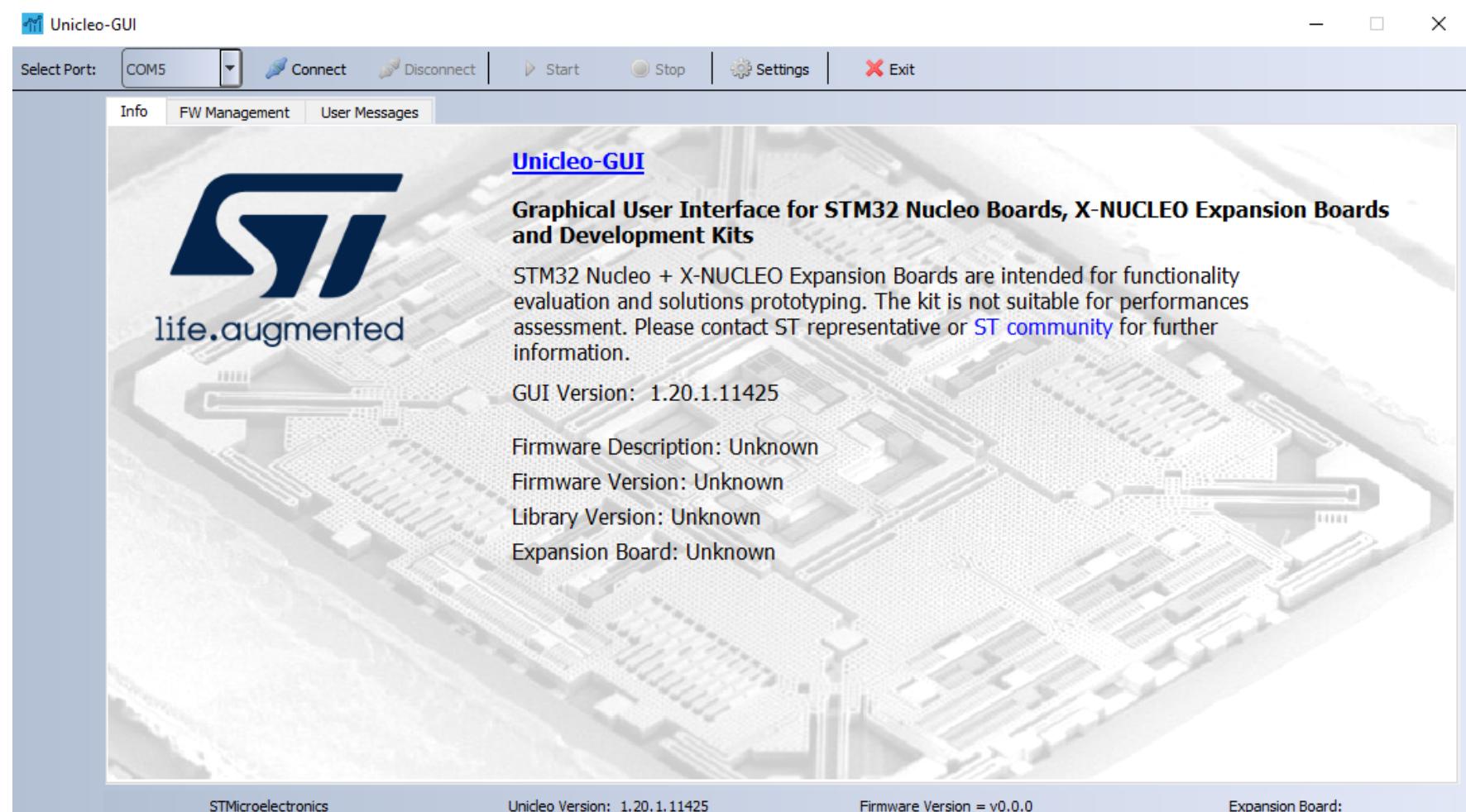
<terminated> DataLogExtended [STM32 C/C++ Application] ST-LINK (ST-LINK GDB server) (Terminated Aug 23, 2022, 11:50:10 AM) [pid: 5]
Download verified successfully

Shutting down...
Exit.

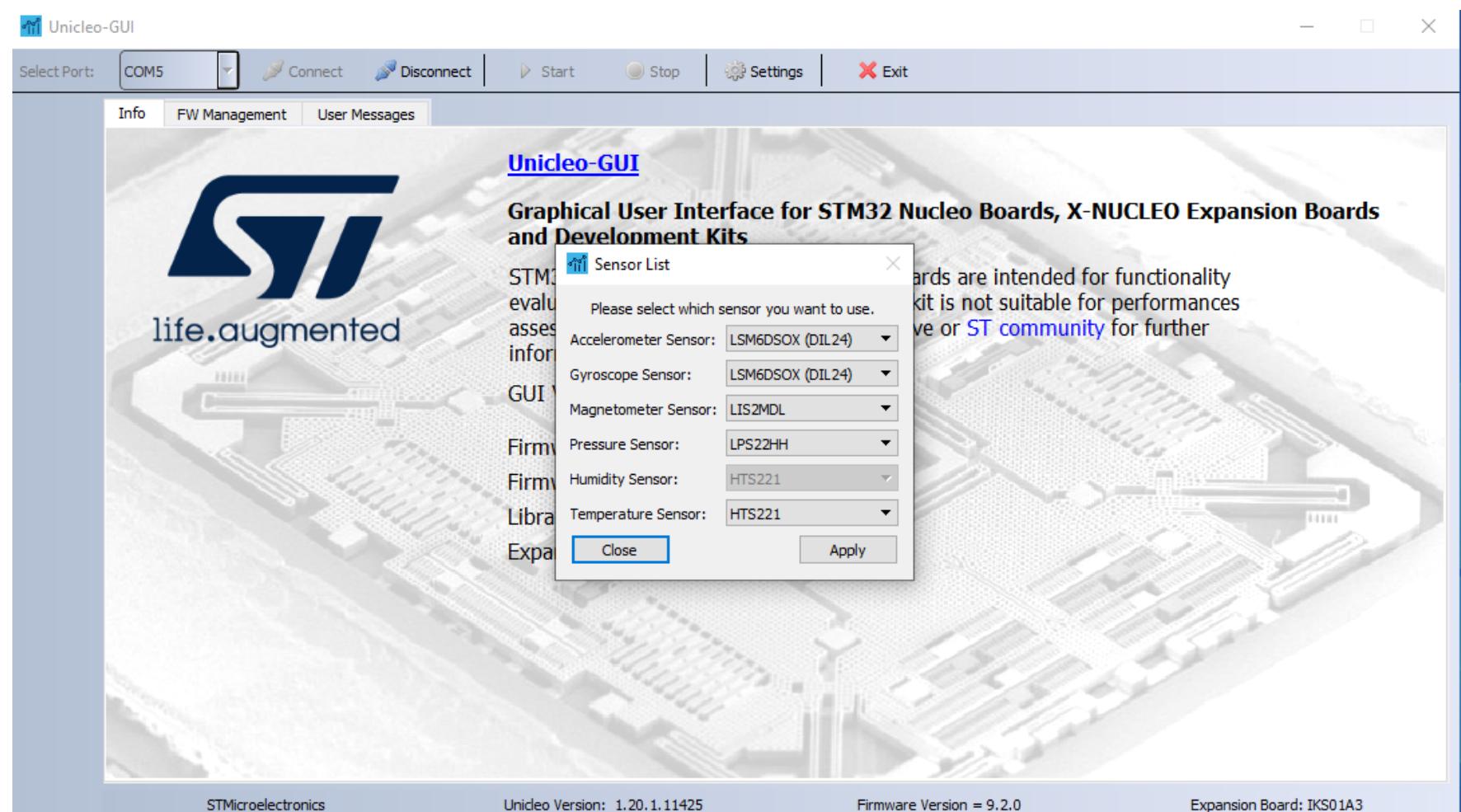
```

- Now let's check if our sensor works as intended in Unicleo.

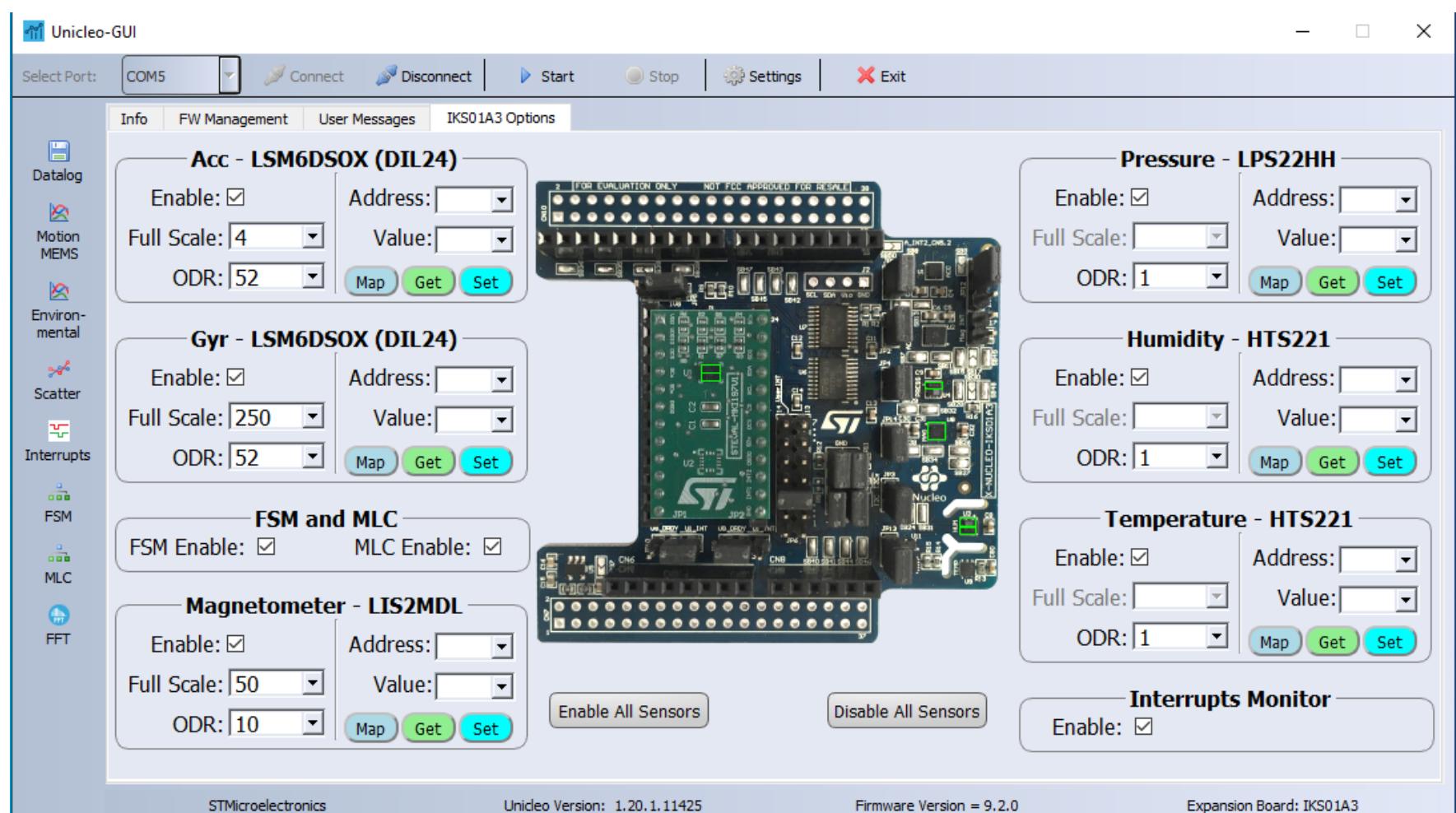
- This is the interface of Unicleo. If our board is connected via USB (ST Link) then the Serial Port should be automatically selected, mine for example is COM5. We select **Connect**.



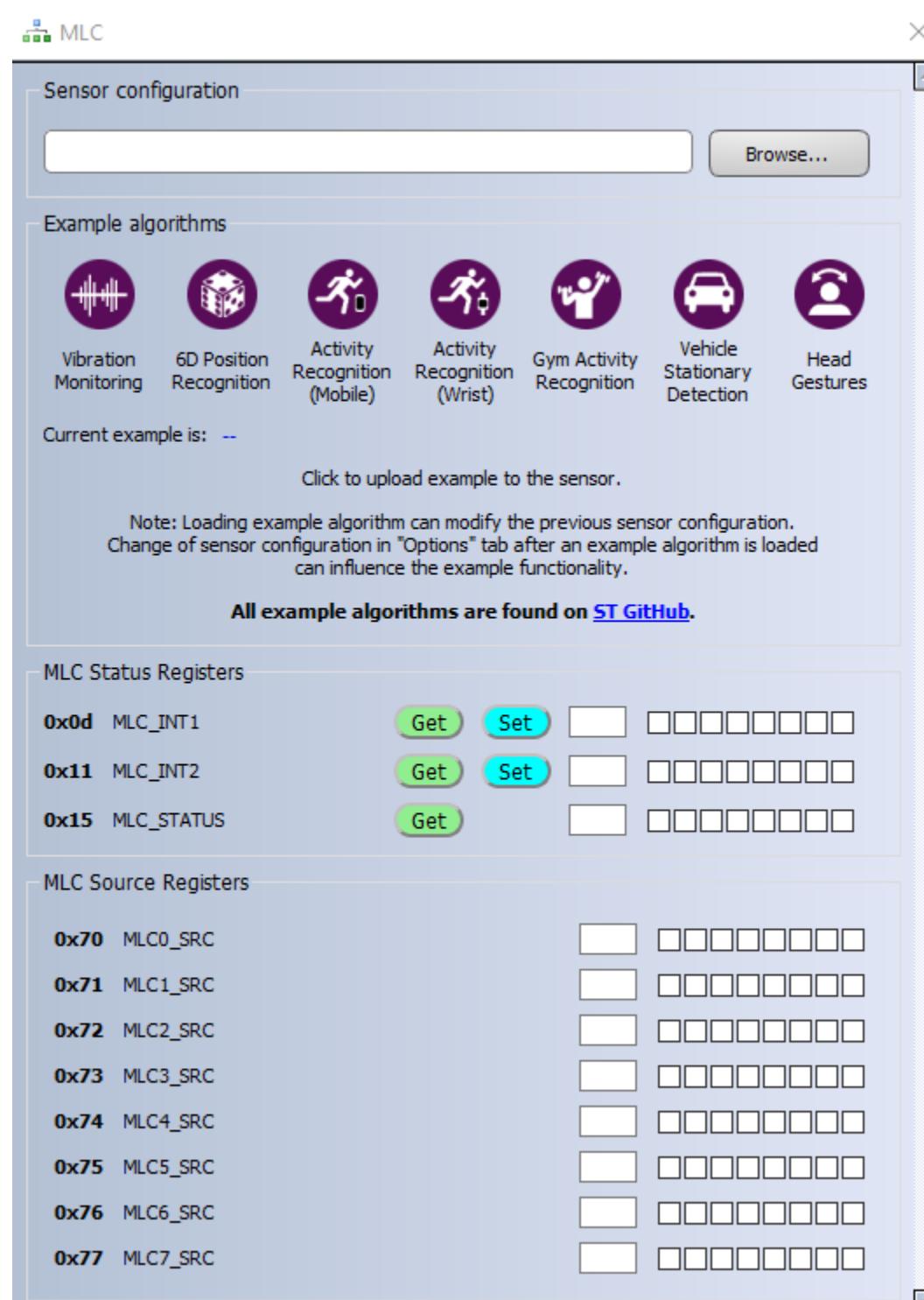
- The sensors list will pop up and show all available sensors. Choose the LSM6DSOX (DIL24).
 - Note that there is also another sensor named LSM6DSO, that is the exact same sensor but on the extender IKS01A3 and does not contain the Machine Learning Core.



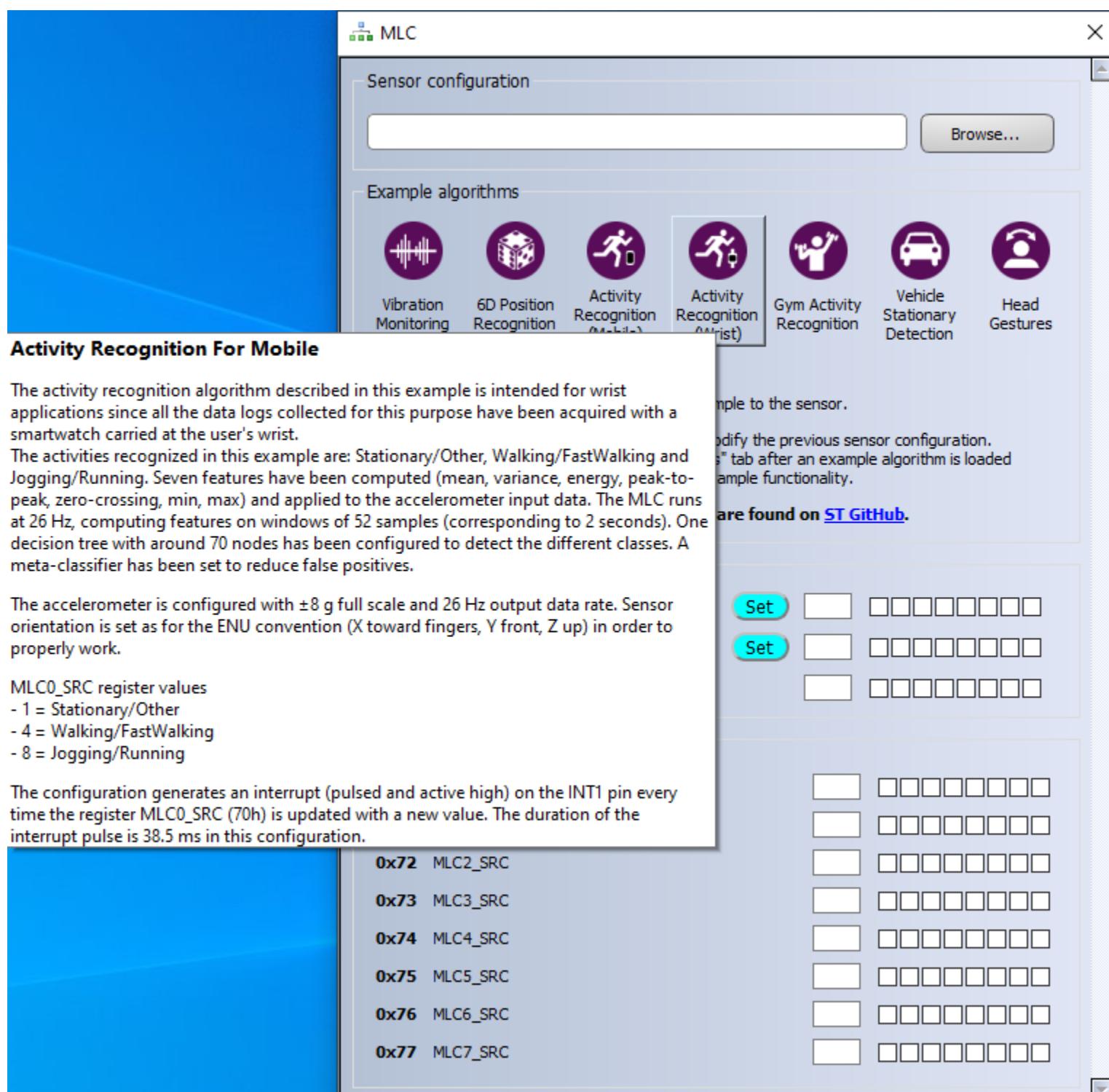
- After **Apply** has been selected, the following window will pop up. It shows the sensors of the IKS01A3 and the MKI197V1 along with their locations.
 - On the left we can see the option for the visualization of the available sensors.



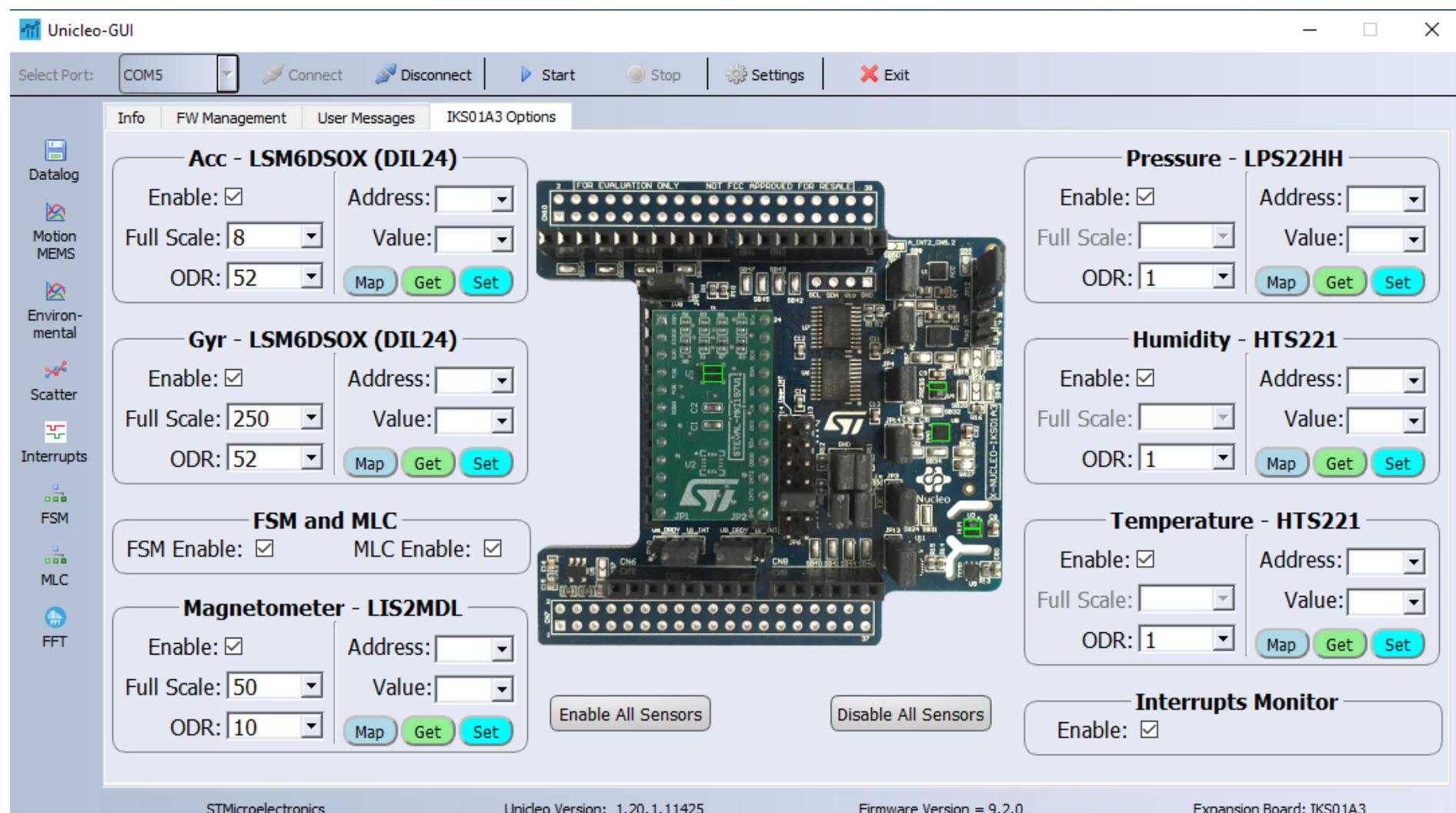
- If **MLC** is chosen on the left, we will be greeted by the following window. On the first block, named **Sensor Configuration**, a custom .ucf dataset can be loaded. Below that are some **Example algorithms** that if chosen they will be loaded and according to the motion of the sensor, a different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



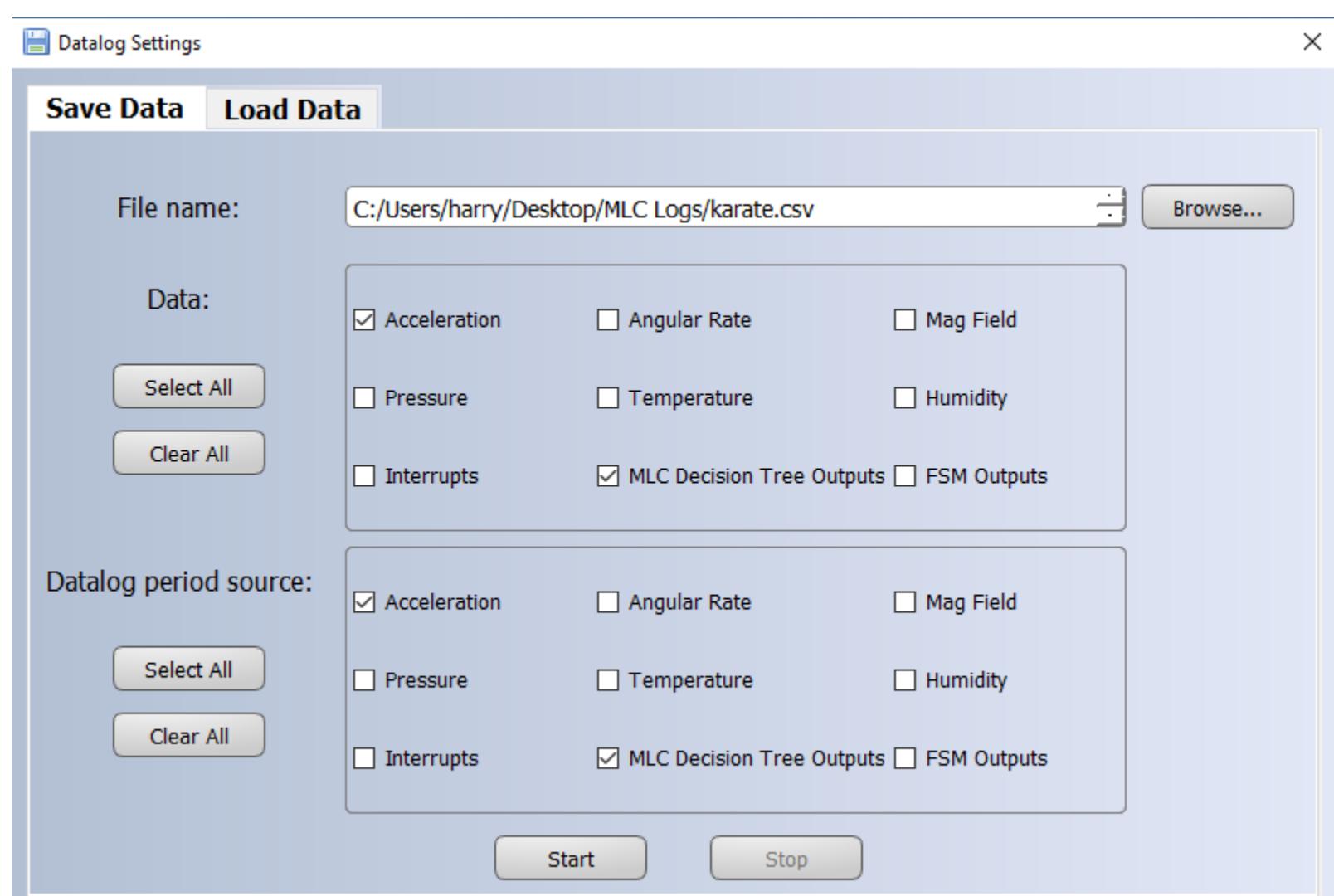
- For example, here are the specifics of the `Activity Recognition (Wrist)` algorithm.
 - According to the documentation
 - 1 = Stationary/Other
 - 4 = Walking/FastWalking
 - 8 = Jogging/Running



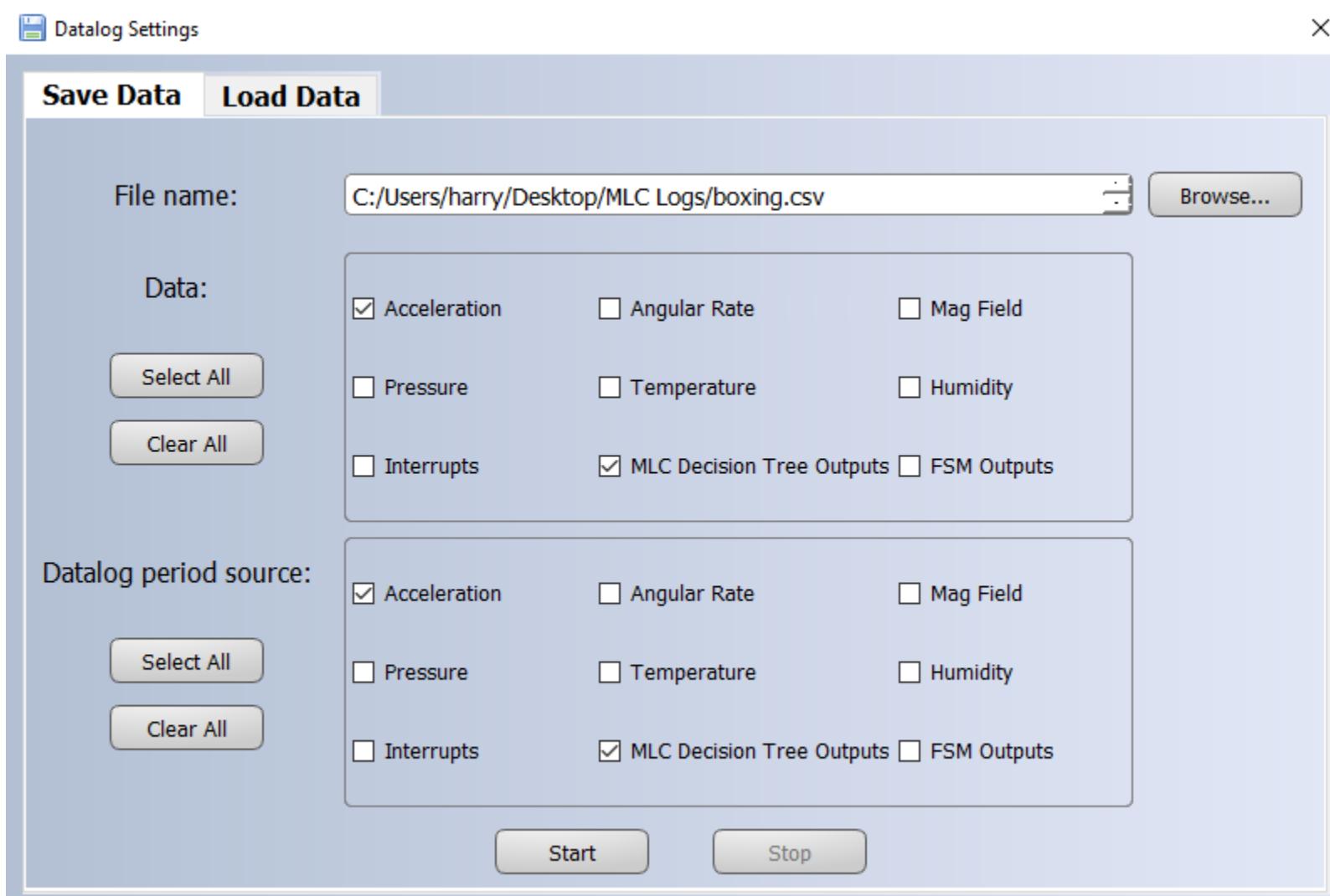
- Before we can select the algorithm we need to select `Start` on the main window. That way we initialize the sensors to begin monitoring.



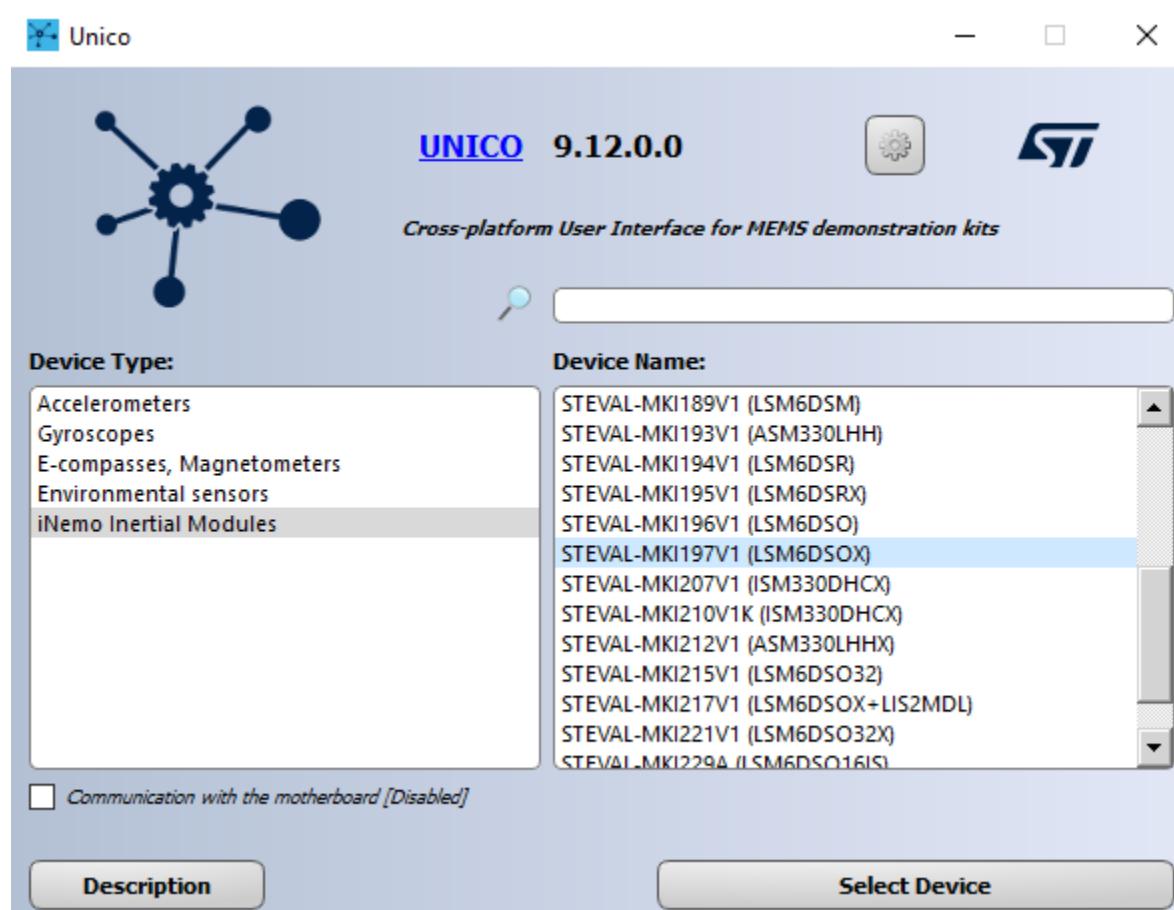
- Now, instead of using one of the example algorithms, I will create my own `.ucf` file. To do that, I need to log data based on each action that I want to add to the Decision Tree. The log will be created in Unicole (in `.txt/.csv` format) and then I'll use Unico to create the `.ucf` file. Then I'll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



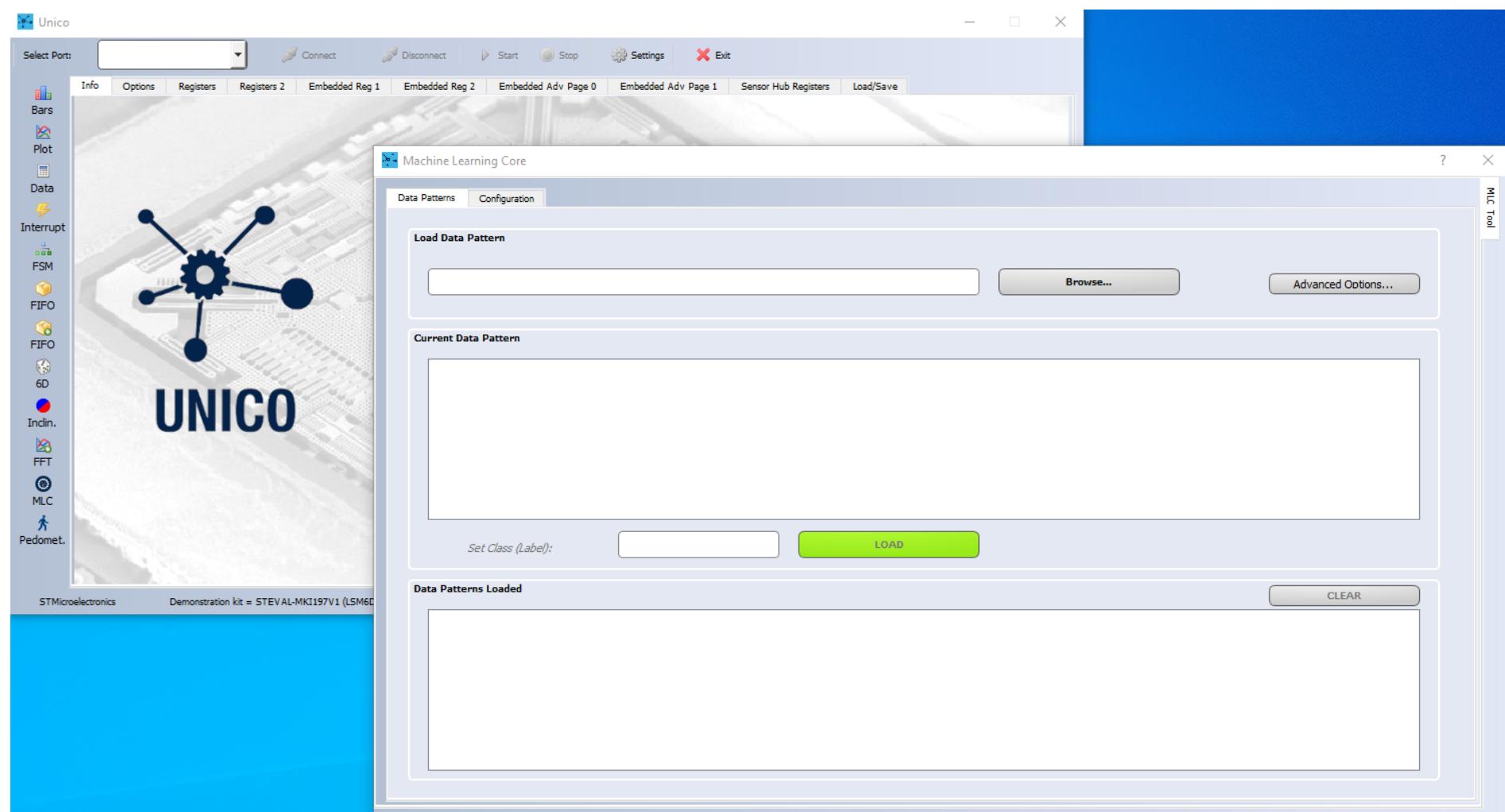
- I will follow the same steps for the dataset `boxing.csv`.



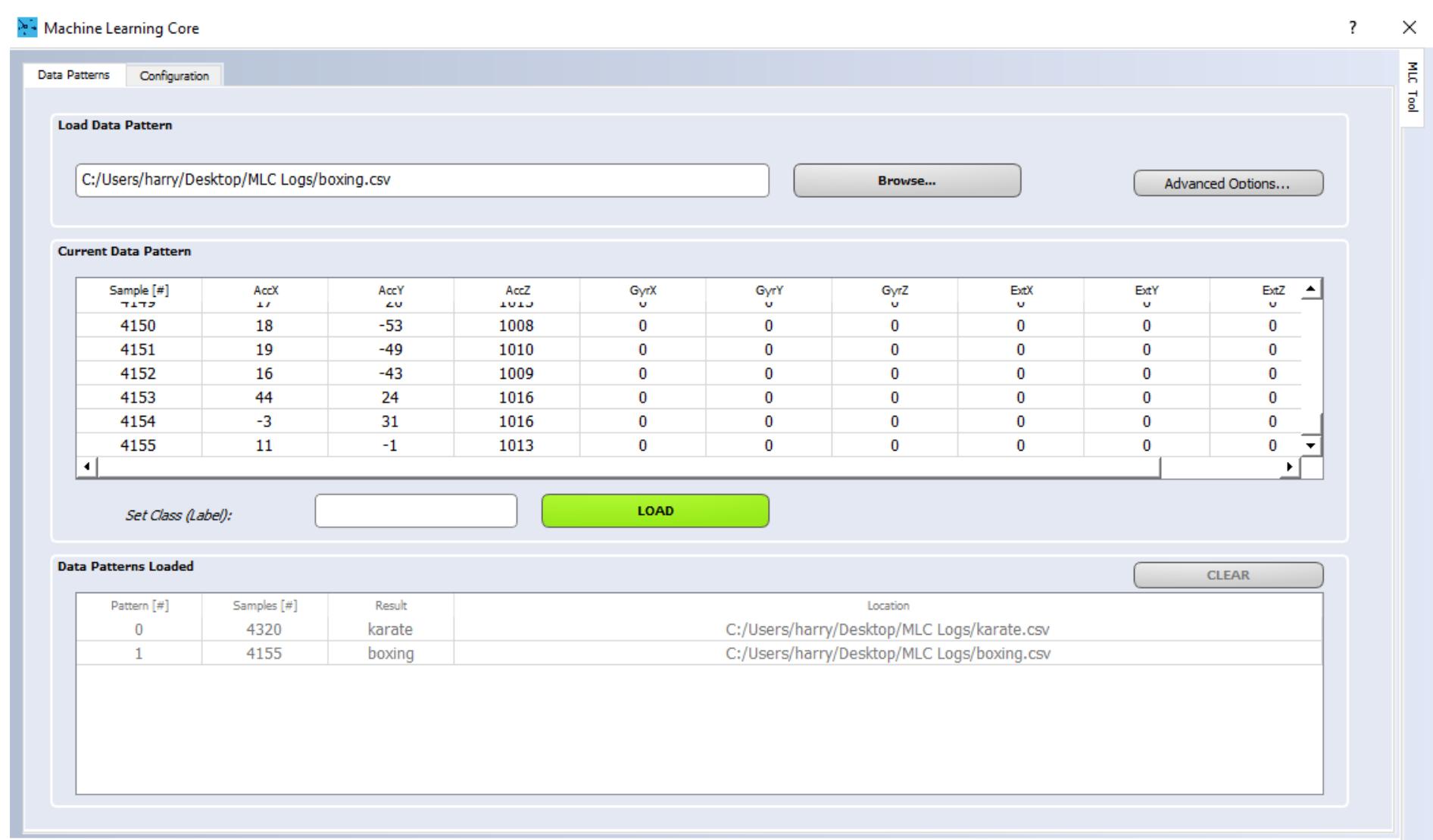
- Now *Unico* needs to be used in order to create a file that can be read by Weka.
 - Since *Unico* cannot be used with the shield IKS01A3, I will have to use it in offline mode. That means that it will not connect with my board, but I will be able to load my datasets in order to extract an `.arff` file.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

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 **Next** **100%**

Machine Learning Core

Machine Learning Core configuration

Decision trees
Number of decision trees: **1**

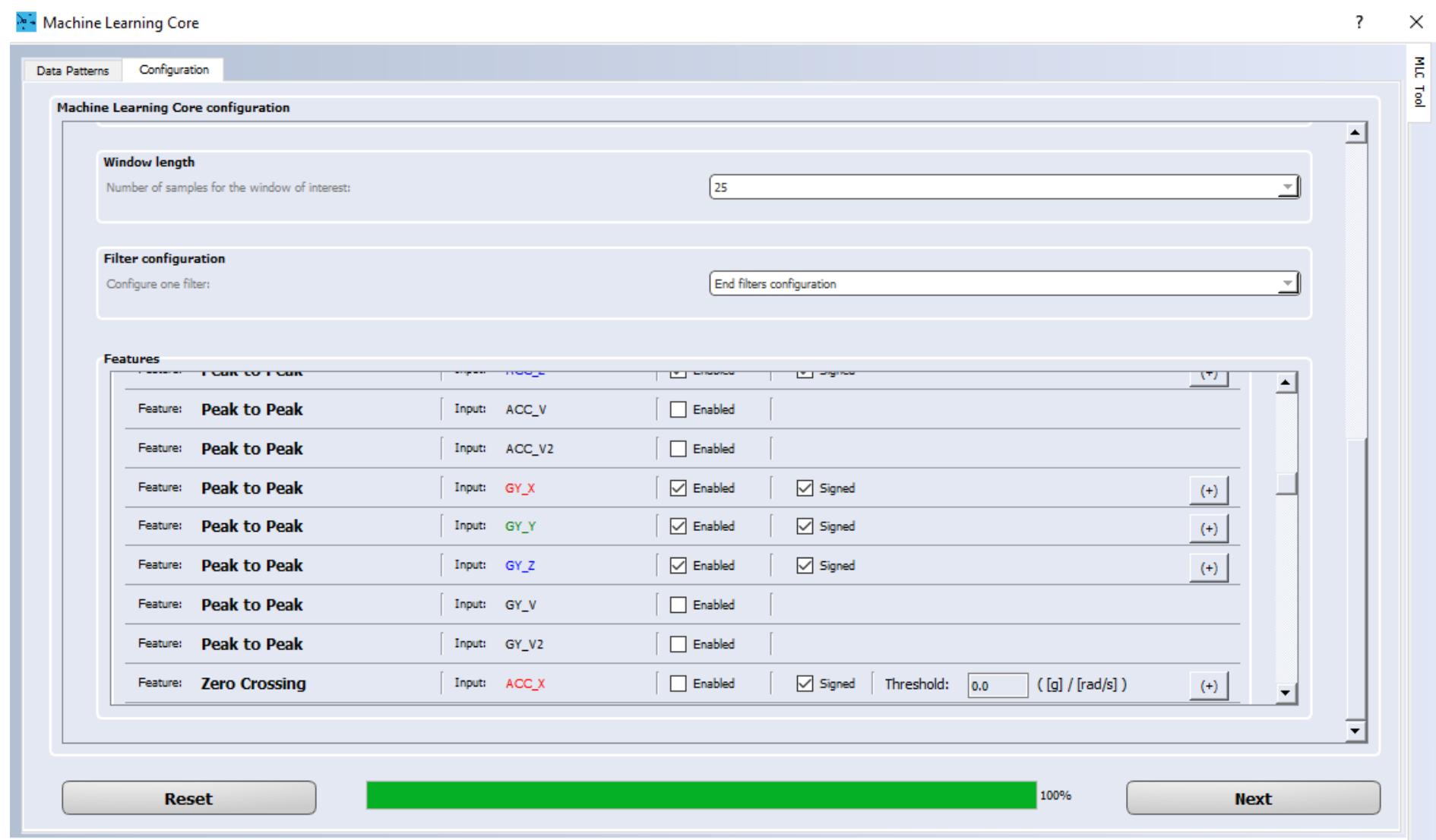
Window length
Number of samples for the window of interest: **25**

Filter configuration
Configure one filter: **End filters configuration**

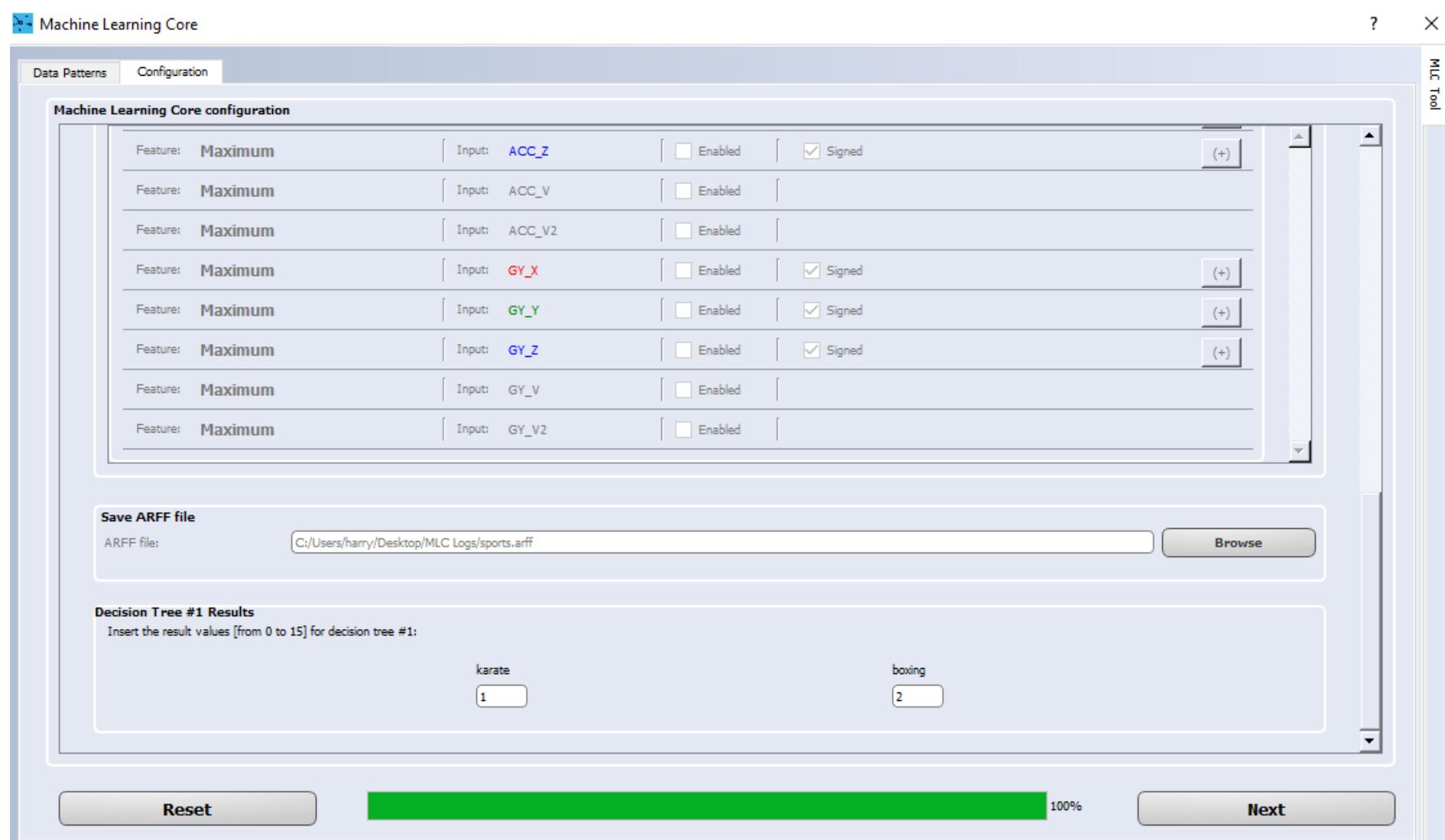
Features

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	(+)

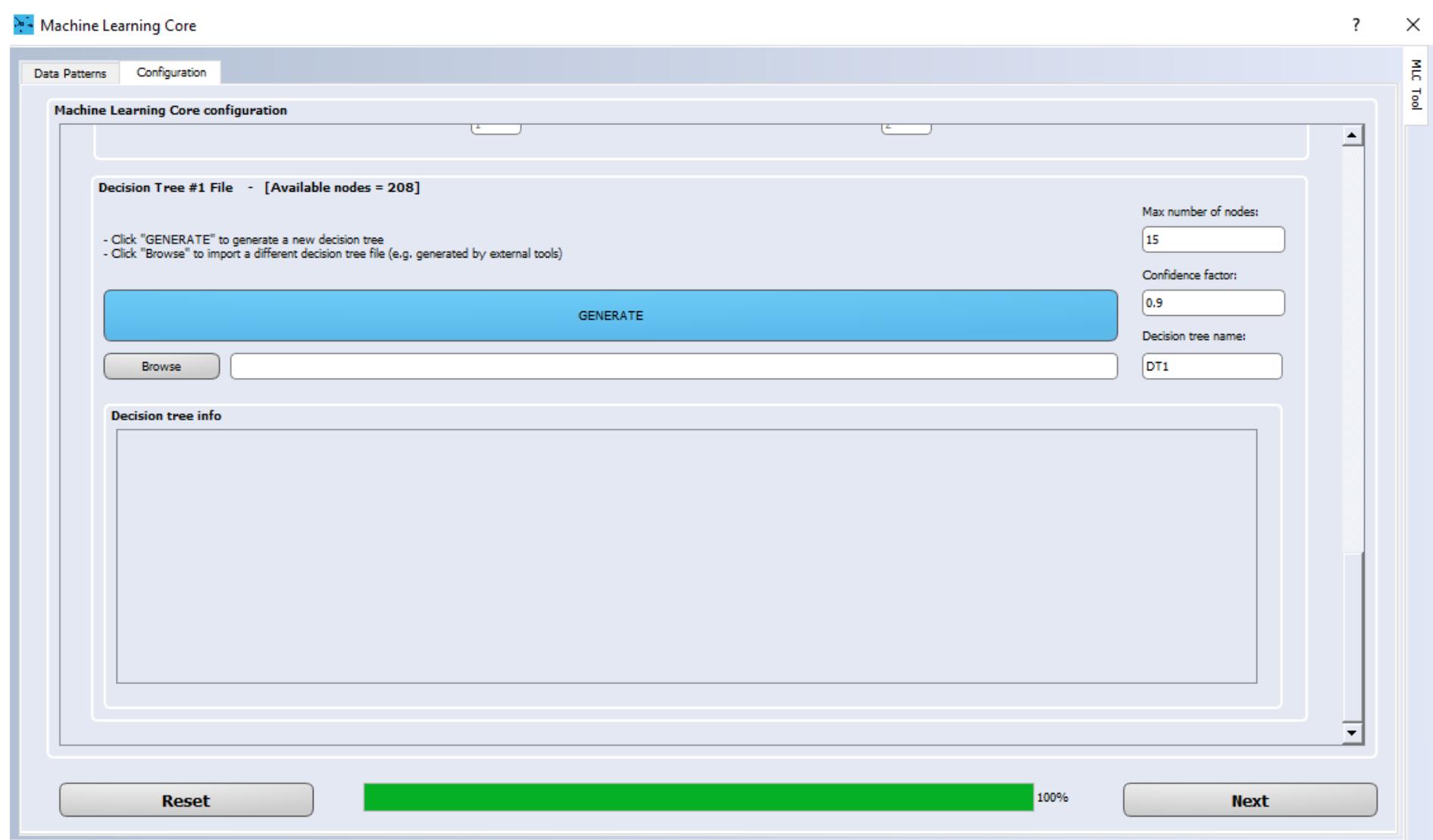
Reset **Next** **100%**



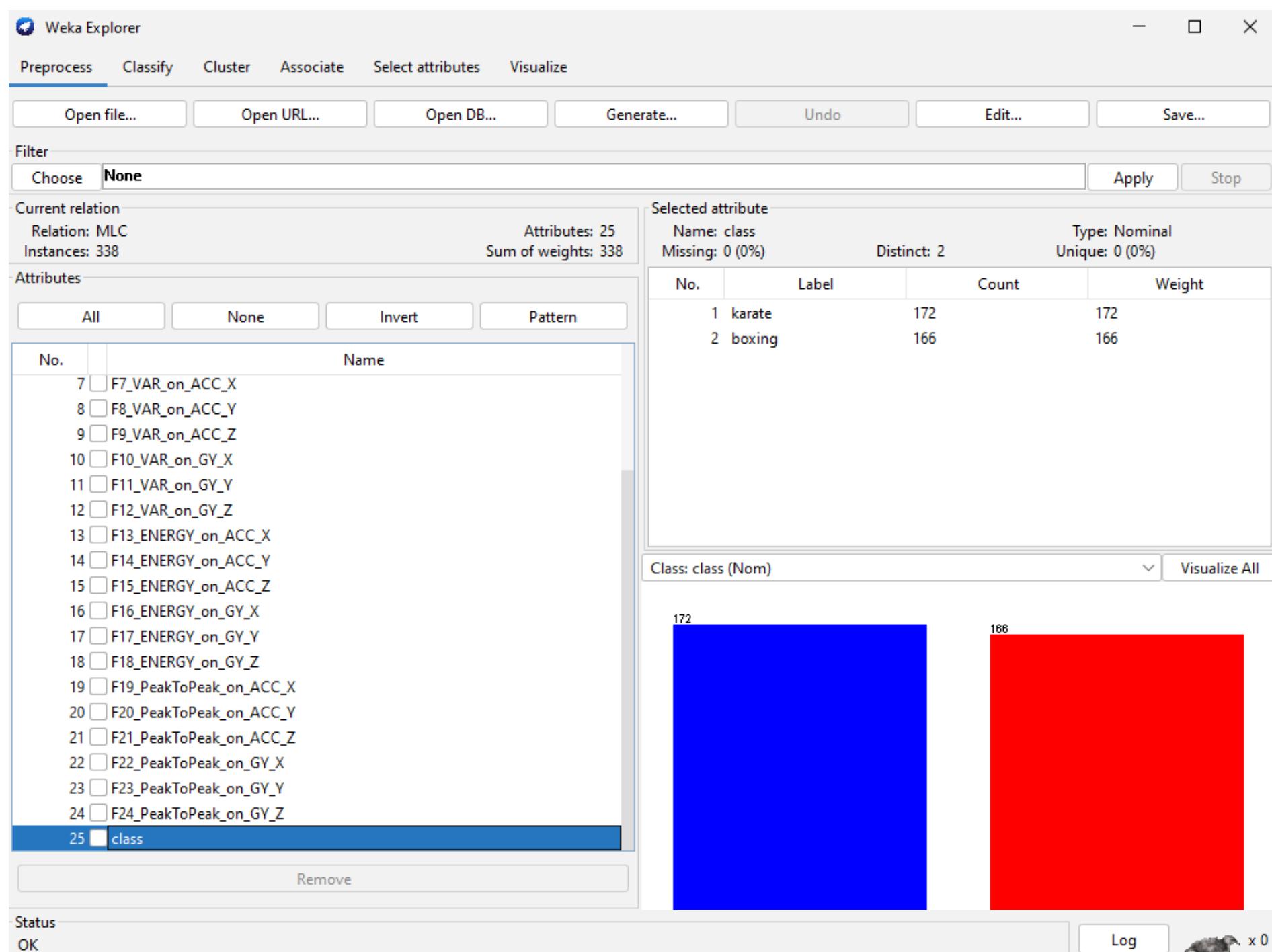
- I chose all the *Signed* options `ACC_X`, `ACC_Y`, `ACC_Z`, `GY_X`, `GY_Y`, `GY_Z` for the `Mean`, `Variance`, `Energy` and `Peak to Peak` features.



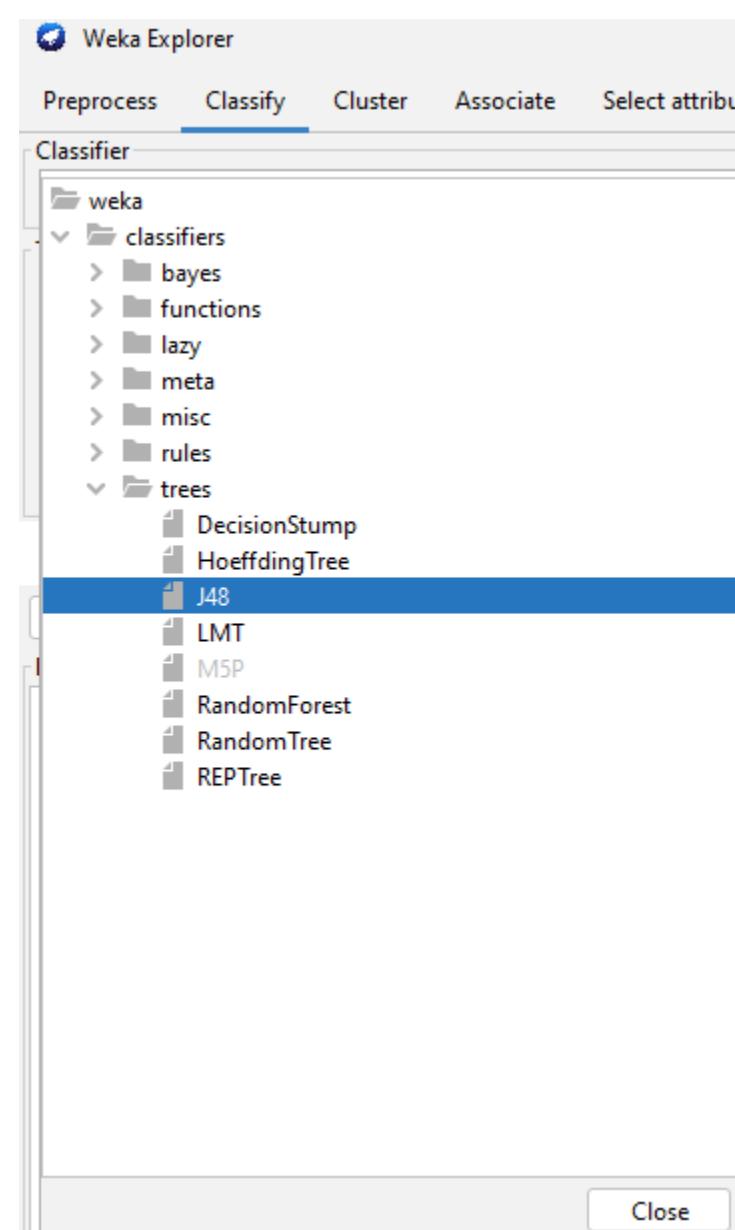
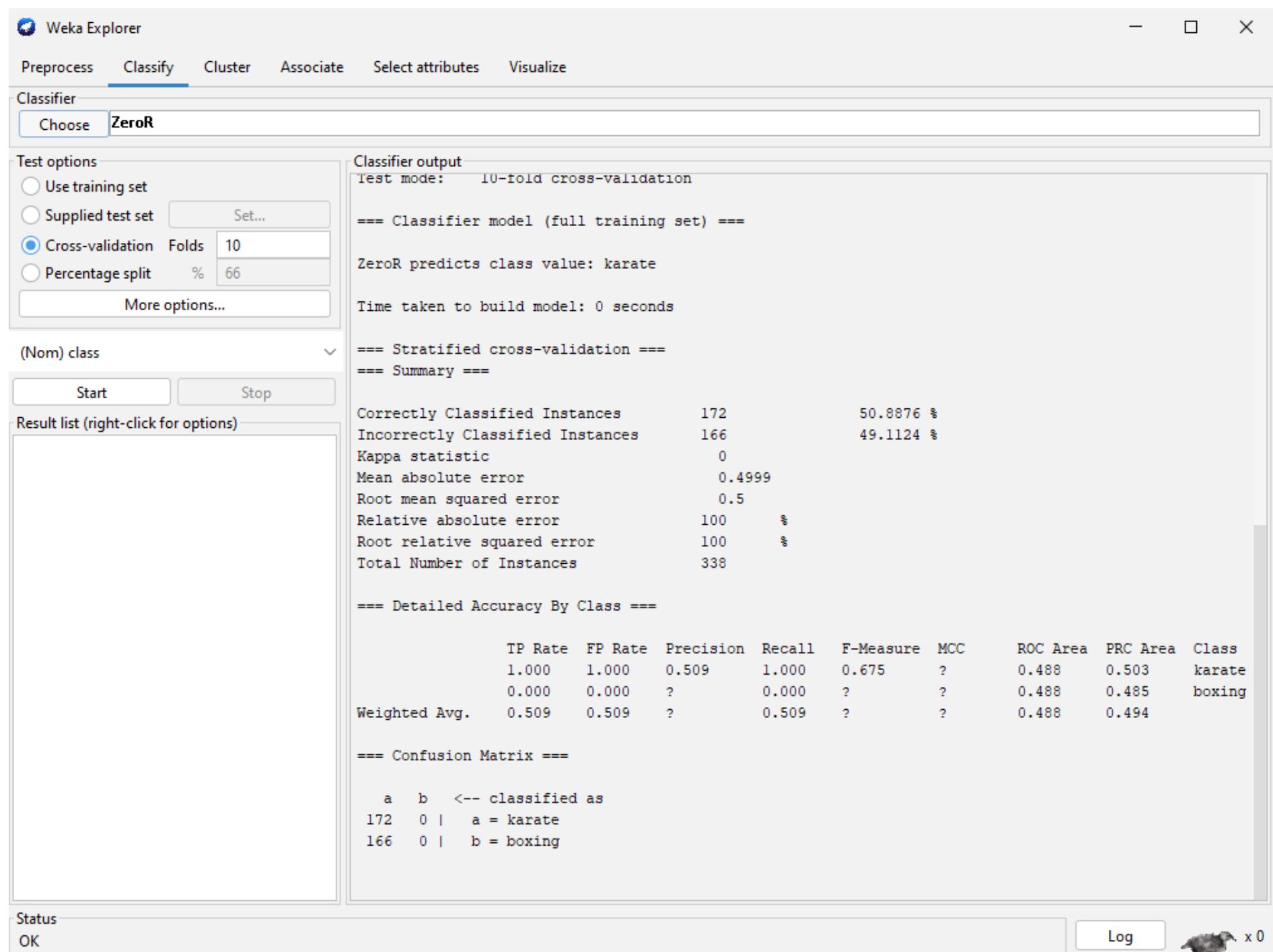
- Here I name a file to save as `.arff` and choose the output that I want to see in the Decision Tree later.
 - I set `1` for karate and `2` for boxing.



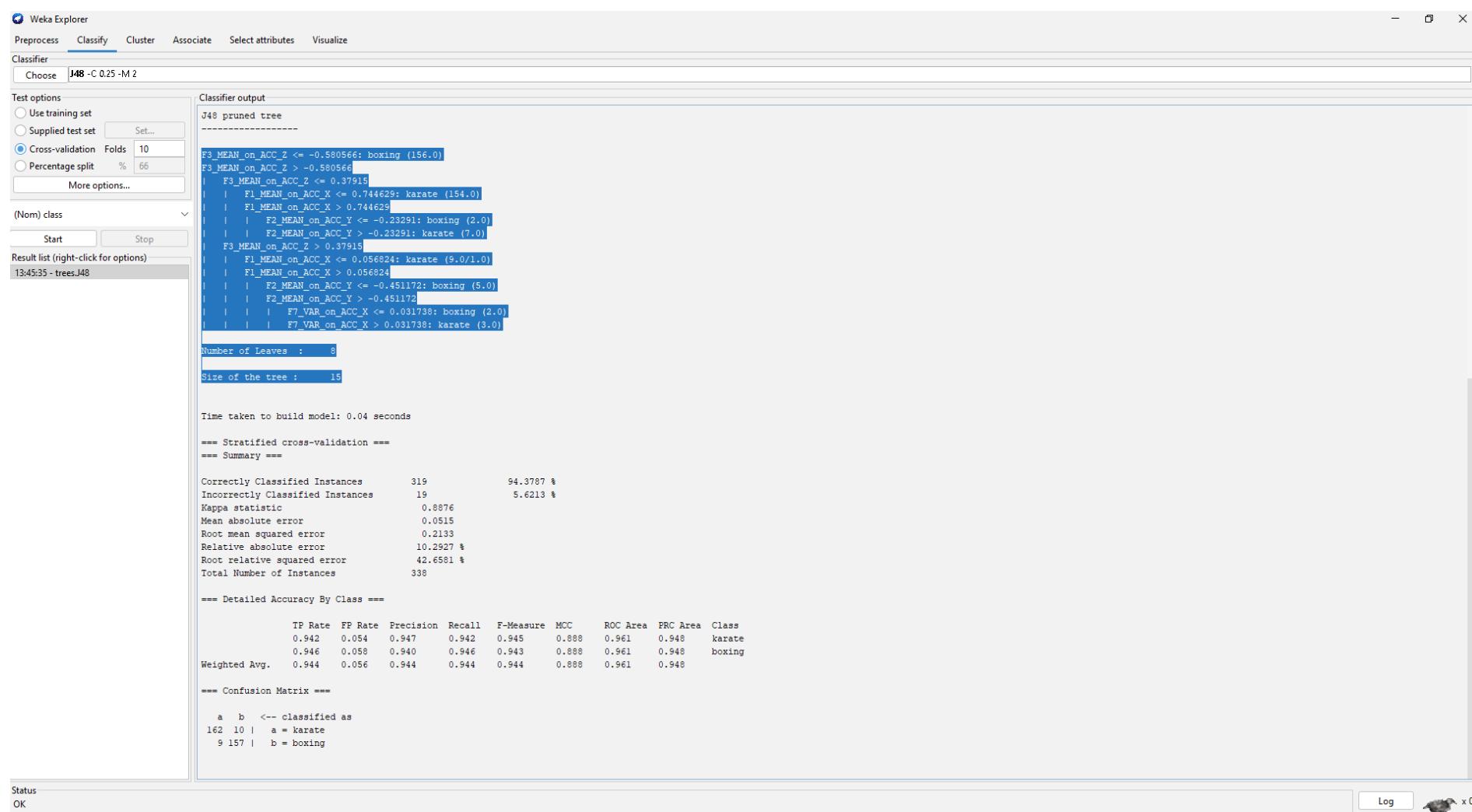
- Open Weka Explorer and load the `sports.arff` file we created above:



- 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.



- The Decision Tree has been 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.



```

Weka Explorer
Preprocess Classify Cluster Associate Select attributes Visualize
Classifier Choose J48 -C 0.25 -M 2
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)
13:45:35 - trees.J48

J48 pruned tree
-----
F3_MEAN_on_ACC_Z <= -0.580566: boxing (156.0)
| F3_MEAN_on_ACC_Z > -0.580566
| | F3_MEAN_on_ACC_Z <= 0.37915
| | | F1_MEAN_on_ACC_X <= 0.744629: karate (154.0)
| | | F1_MEAN_on_ACC_X > 0.744629
| | | | F2_MEAN_on_ACC_Y <= -0.23291: boxing (2.0)
| | | | F2_MEAN_on_ACC_Y > -0.23291: karate (7.0)
| | | F3_MEAN_on_ACC_Z > 0.37915
| | | | F1_MEAN_on_ACC_X <= 0.056824: karate (9.0/1.0)
| | | | F1_MEAN_on_ACC_X > 0.056824
| | | | | F2_MEAN_on_ACC_Y <= -0.451172: boxing (5.0)
| | | | | F2_MEAN_on_ACC_Y > -0.451172
| | | | | | F7_VAR_on_ACC_X <= 0.031738: boxing (2.0)
| | | | | | F7_VAR_on_ACC_X > 0.031738: karate (3.0)

Number of Leaves : 8
Size of the tree : 15

Time taken to build model: 0.04 seconds

==== Stratified cross-validation ====
==== Summary ====
Correctly Classified Instances 319 94.3787 %
Incorrectly Classified Instances 19 5.6213 %
Kappa statistic 0.8876
Mean absolute error 0.0515
Root mean squared error 0.2133
Relative absolute error 10.2927 %
Root relative squared error 42.6581 %
Total Number of Instances 338

==== Detailed Accuracy By Class ====

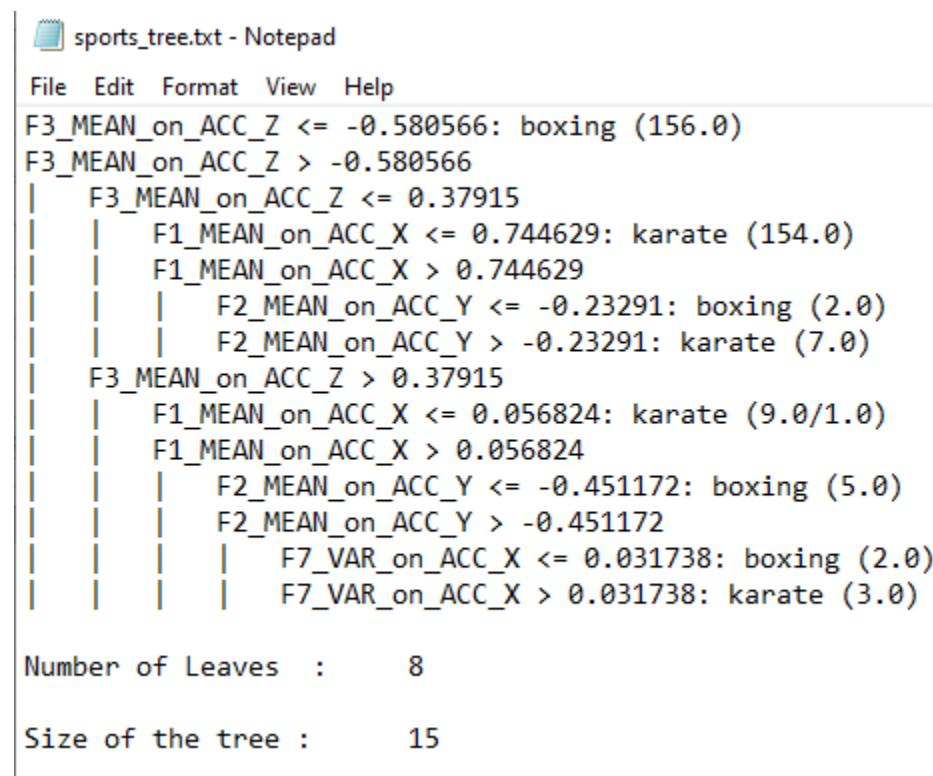

|               | TP Rate | FP Rate | Precision | Recall | F-Measure | MCC   | ROC Area | PRC Area | Class  |
|---------------|---------|---------|-----------|--------|-----------|-------|----------|----------|--------|
| 0.942         | 0.054   | 0.947   | 0.942     | 0.945  | 0.988     | 0.961 | 0.948    | 0.948    | karate |
| 0.946         | 0.058   | 0.940   | 0.946     | 0.943  | 0.988     | 0.961 | 0.948    | 0.948    | boxing |
| Weighted Avg. | 0.944   | 0.056   | 0.944     | 0.944  | 0.944     | 0.988 | 0.961    | 0.948    |        |



==== Confusion Matrix ====


|  |  | a   | b   | <-- classified as |
|--|--|-----|-----|-------------------|
|  |  | 162 | 10  | a = karate        |
|  |  | 9   | 157 | b = boxing        |


```



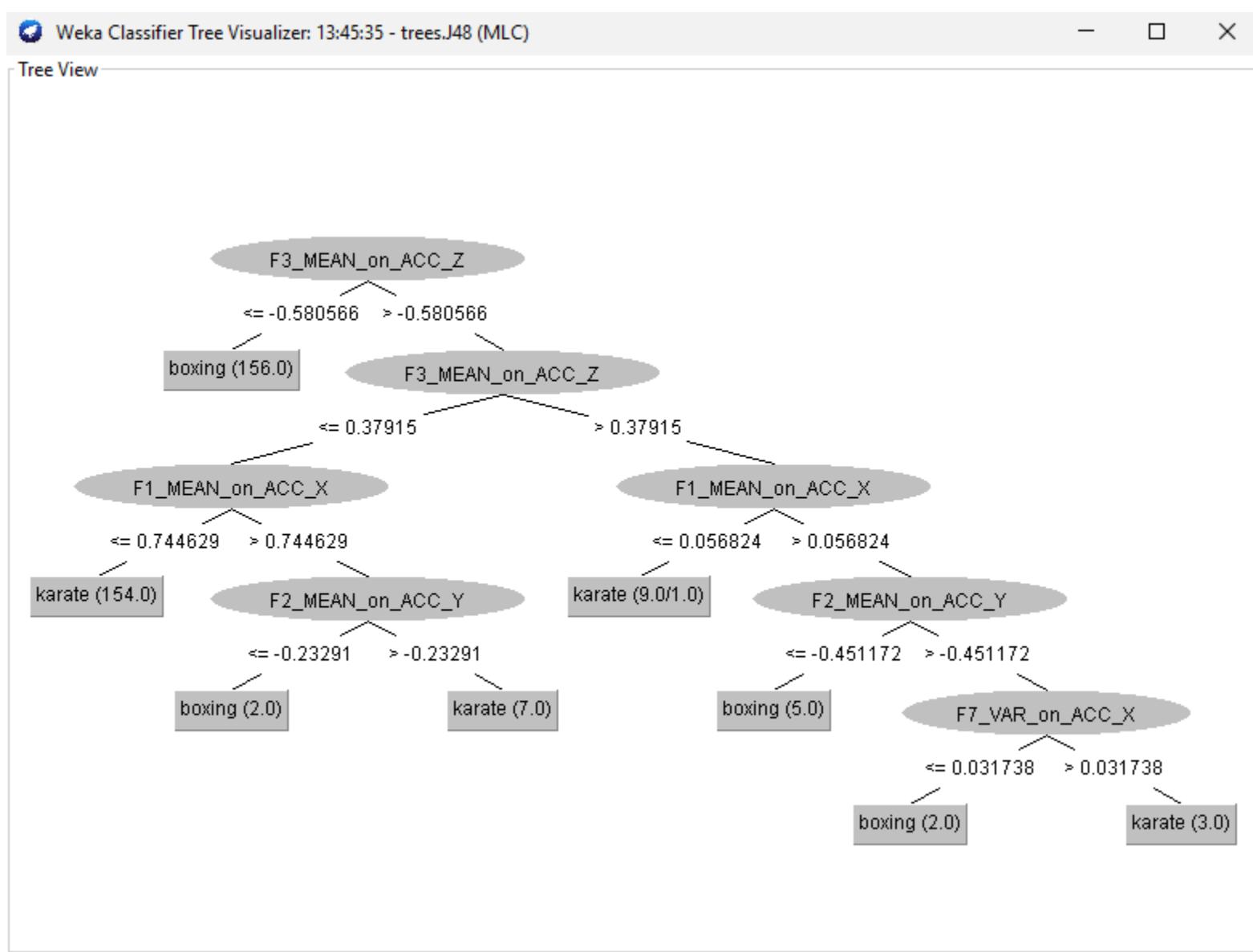
```

sports_tree.txt - Notepad
File Edit Format View Help
F3_MEAN_on_ACC_Z <= -0.580566: boxing (156.0)
F3_MEAN_on_ACC_Z > -0.580566
| F3_MEAN_on_ACC_Z <= 0.37915
| | F1_MEAN_on_ACC_X <= 0.744629: karate (154.0)
| | F1_MEAN_on_ACC_X > 0.744629
| | | F2_MEAN_on_ACC_Y <= -0.23291: boxing (2.0)
| | | F2_MEAN_on_ACC_Y > -0.23291: karate (7.0)
| | F3_MEAN_on_ACC_Z > 0.37915
| | | F1_MEAN_on_ACC_X <= 0.056824: karate (9.0/1.0)
| | | F1_MEAN_on_ACC_X > 0.056824
| | | | F2_MEAN_on_ACC_Y <= -0.451172: boxing (5.0)
| | | | F2_MEAN_on_ACC_Y > -0.451172
| | | | | F7_VAR_on_ACC_X <= 0.031738: boxing (2.0)
| | | | | F7_VAR_on_ACC_X > 0.031738: karate (3.0)

Number of Leaves : 8
Size of the tree : 15

```

- 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.

Machine Learning Core

Data Patterns Configuration

Machine Learning Core configuration

Decision Tree #1 File - [Available nodes = 208]

- Click "GENERATE" to generate a new decision tree
- Click "Browse" to import a different decision tree file (e.g. generated by external tools)

GENERATE

Browse C:/Users/harry/Desktop/MLC Report/sports_tree.txt.txt

Decision tree info

```

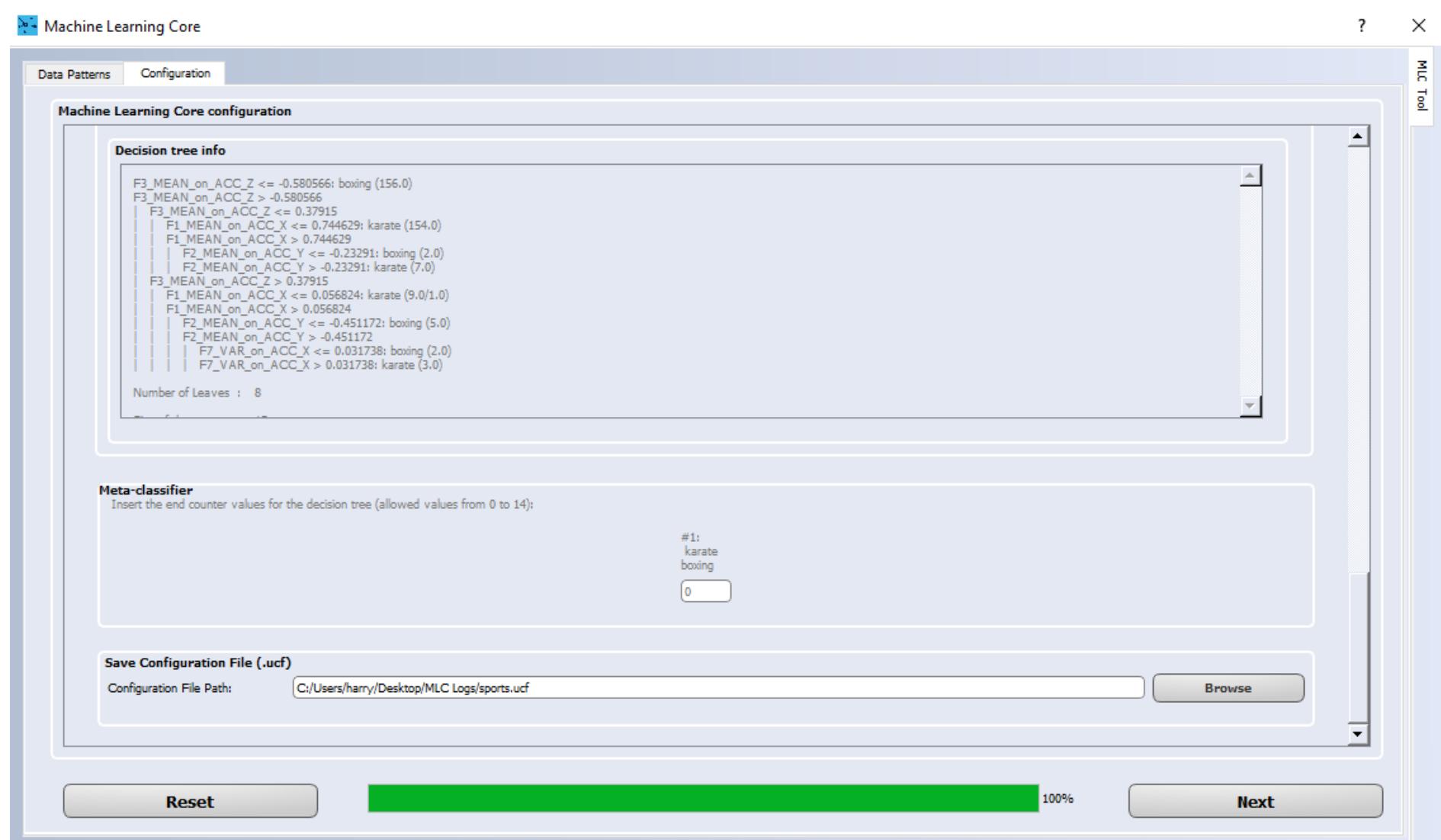
F3_MEAN_on_ACC_Z <= -0.580566: boxing (156.0)
F3_MEAN_on_ACC_Z <= 0.37915:
| F1_MEAN_on_ACC_X <= 0.744629: karate (154.0)
| F1_MEAN_on_ACC_X > 0.744629:
| | F2_MEAN_on_ACC_Y <= -0.23291: boxing (2.0)
| | F2_MEAN_on_ACC_Y > -0.23291: karate (7.0)
F3_MEAN_on_ACC_Z > 0.37915:
| F1_MEAN_on_ACC_X <= 0.056824: karate (9.0/1.0)
| F1_MEAN_on_ACC_X > 0.056824:
| | F2_MEAN_on_ACC_Y <= -0.451172: boxing (5.0)
| | F2_MEAN_on_ACC_Y > -0.451172:
| | | F7_VAR_on_ACC_X <= 0.031738: boxing (2.0)
| | | F7_VAR_on_ACC_X > 0.031738: karate (3.0)
  
```

Number of Leaves : 8

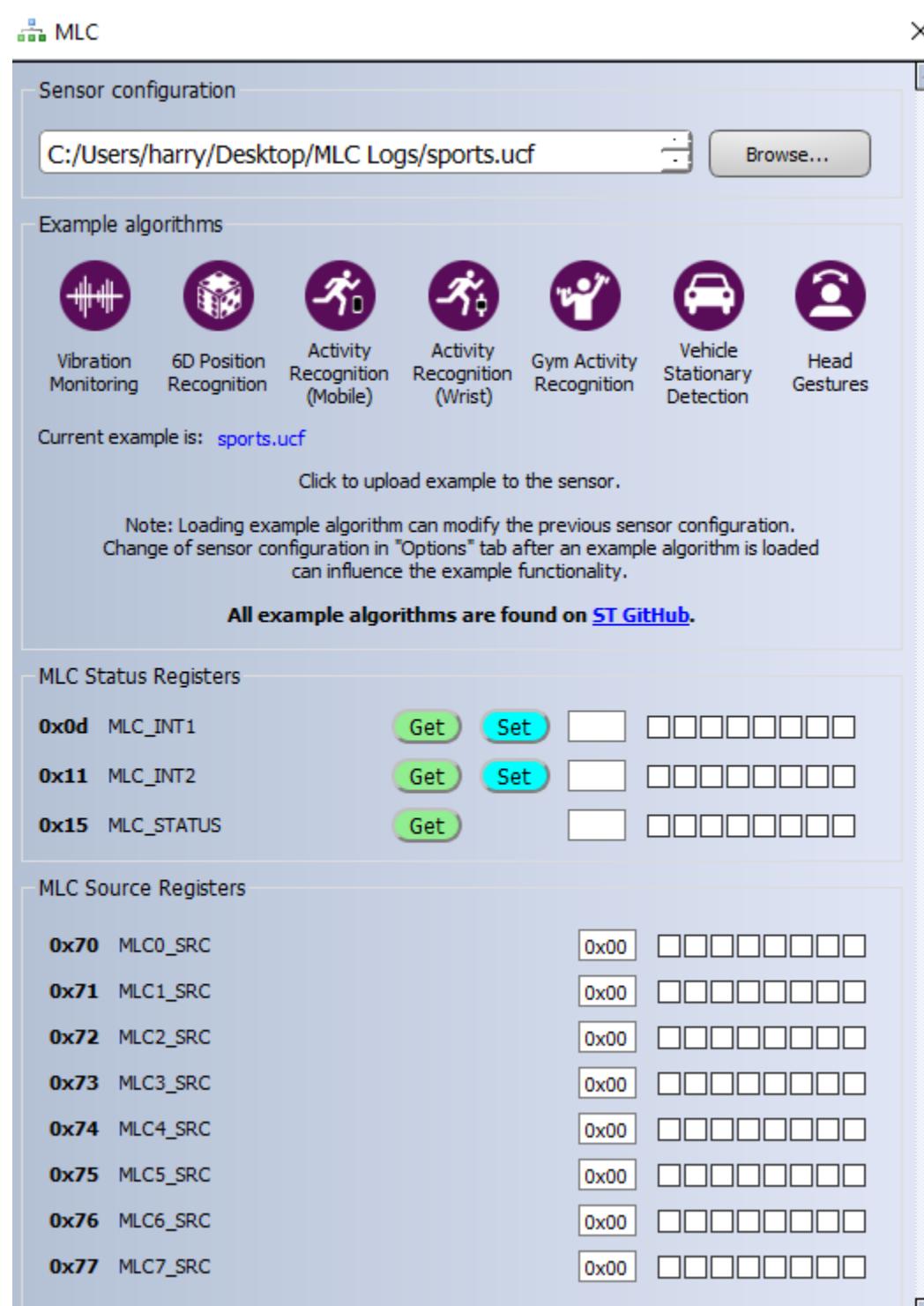
Max number of nodes: 15
Confidence factor: 0.9
Decision tree name: DT1

Reset **Next**

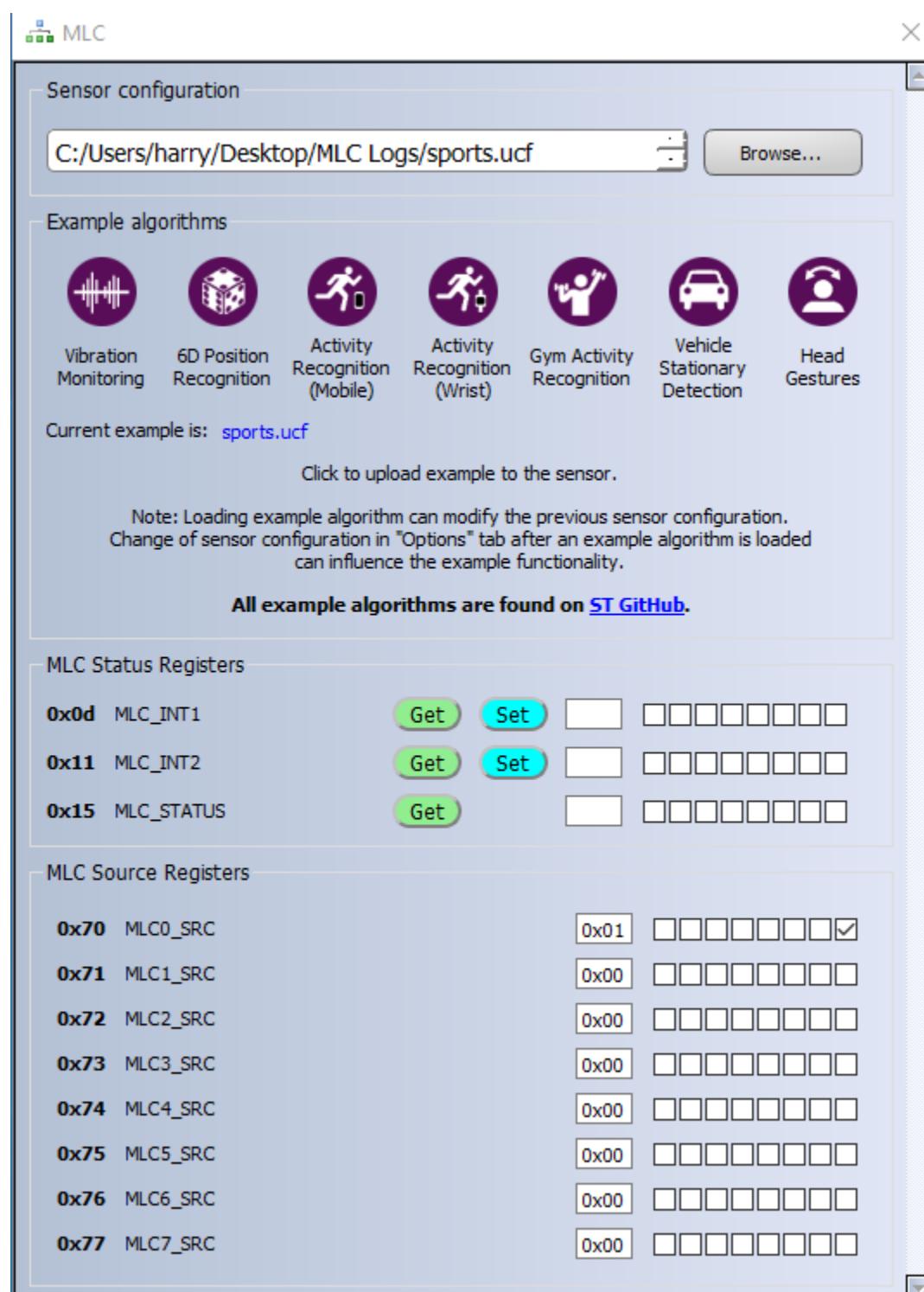
- And save the file as `sports.ucf`.



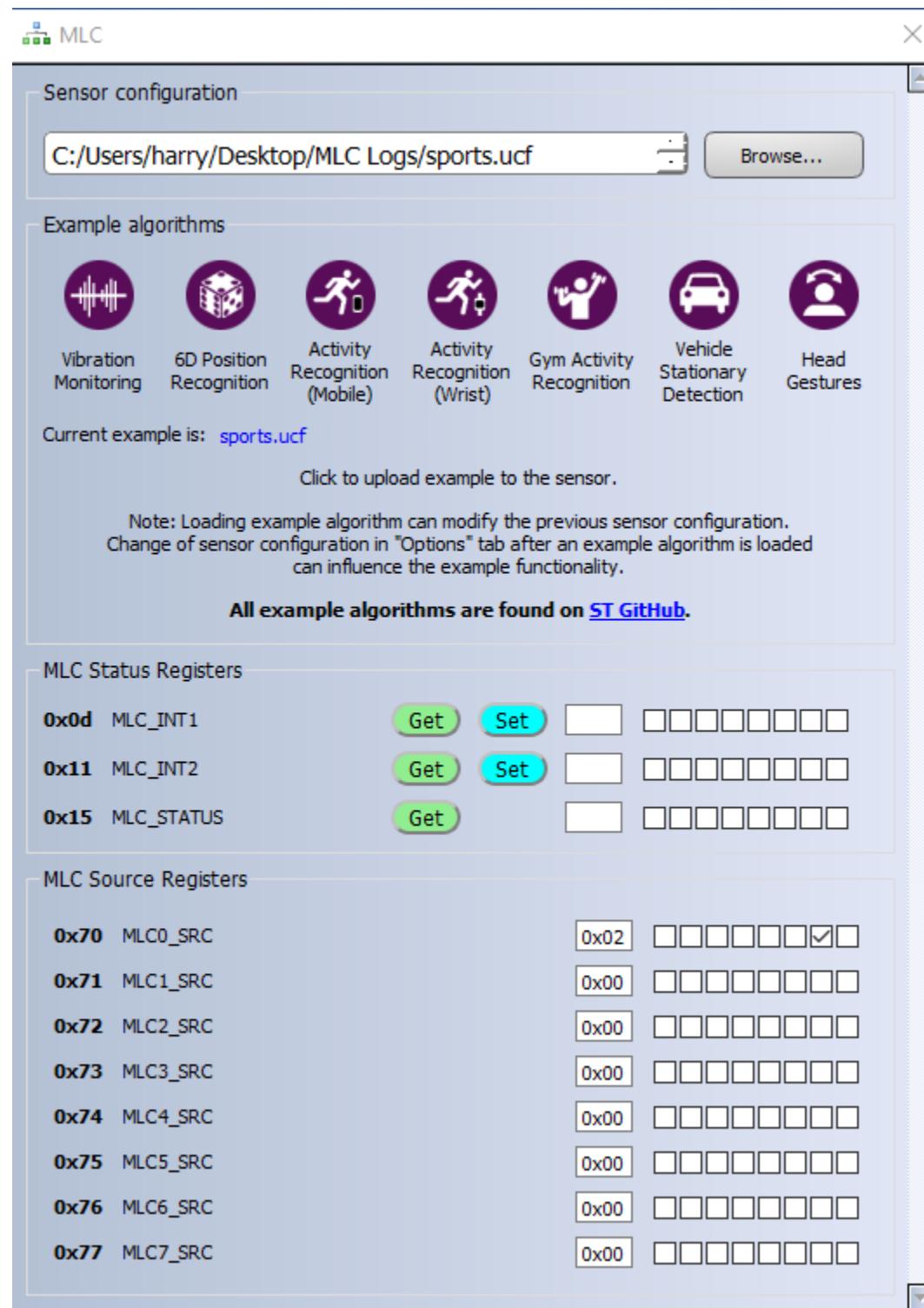
- Now we can go back to Unicleo and load the `sports.ucf` file to the `MLC`.



- 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 demonstration 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 Unicrelo/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.