



Application Software Pack: ML-based State Monitor for FXLS8974CF

Revision 9
January 2024

Contents

1 Lab Overview	3
2 Software and Hardware Installation	3
2.1 Hardware Requirements	3
2.2 TensorFlow Installation	4
2.3 Download Enablement Software	4
2.4 eIQ Tool Installation	4
3 Import the App Software Pack into MCUXpresso IDE	4
3.1 Option #1: Get the App Software Pack with MCUXpresso IDE	4
3.2 Option #2: Use the Command Line	6
4 Gather the Data	7
4.1 Import the ML-based System State Monitor project	7
4.2 Collect data	8
5 Train Data and create a model	11
5.1 Copy Training Data to PC	11
5.2 Train The Model	12
6 Update Model for Inference Engines	14
6.1 Update Model Configuration File	14
6.2 TensorFlow Lite for Microcontrollers	15
6.3 Glow	15
7 Deploy Model To Board	17
8 Conclusion	19

1 Lab Overview

This lab will cover how to use the ML-based fan state monitor application code in conjunction with the [Building and benchmarking Deep Learning models for Smart Sensing Appliances on MCUs application note \(AN13562\)](#). This document will cover how to gather the data that the model will be trained on, how to run the Jupyter notebook scripts that will train the model, and how to deploy the trained model to the board.

This lab was written for the i.MX RT1170, but the same steps can be used for the following boards:

- [LPCXpresso55S69](#)
- [i.MX RT1170-EVK](#)

This lab document is for the [FXLS8974CF](#) accelerometer which can be found on the following sensor boards:

- [ACCEL_4_CLICK](#) board combined with [Arduino UNO click shield interposer](#) board.
- [FRDM-STBI-A8974](#) sensor board.

The software for this accelerometer uses a new updated version of the application code that includes the ability to evaluate several different time series models like CNN, LightCNN, LightResNet, and MLP. If using the accelerometer on the EVK or the [FRDM-STBC-AGM01](#) board please refer to the other lab document as those use the FXOS8700CQ accelerometer.

2 Software and Hardware Installation

This section will cover the steps needed to install the eIQ software and TensorFlow on your computer.

2.1 Hardware Requirements

- i.MX RT1170 EVK board (or other supported board like LPC55S69)
- A board with the FXLS8974CF accelerometer:
 - [ACCEL_4_CLICK](#) board combined with [Arduino UNO click shield interposer](#) board.



- [FRDM-STBI-A8974](#) sensor board.
 - The signal connected to SH38 needs to be cut to isolate it to prevent the accelerometer from being put into an incorrect mode.



- microSD card (formatted for FAT32)
- microSD card reader to transfer data to a PC
- A fan, motor, or other vibration source to train the model on that can provide vibrations that will be gathered by the sensor. This [fan](#) for example.

2.2 TensorFlow Installation

1. Download and install Python 3.10. ****The 64-bit edition is required and it is highly recommended to use Python 64-bit 3.10.x to ensure compatibility with this lab****:
<https://www.python.org/downloads/>
2. Open a Windows command prompt and verify that the python command corresponds to Python 3.10.x.
python -V
3. Update the python installer tools:
python -m pip install -U pip
python -m pip install -U setuptools
4. Install the Tensorflow libraries and support for python. To ensure software compatibility, it should be these version numbers:
python -m pip install tensorflow==2.8
python -m pip install keras==2.8
python -m pip install protobuf==3.20
python -m pip install pandas==1.5.3
5. Install other useful python packages. Not all of these will be used for this lab but will be useful for other eIQ demos and scripts.
python -m pip install numpy scipy matplotlib ipython jupyter sympy nose imageio
python -m pip install netron seaborn west pyserial scikit-learn opencv-python pillow

2.3 Download Enablement Software

1. Install the latest version of [MCUXpresso IDE](#)
2. Install a terminal program like [TeraTerm](#).
3. Install [Git](#)

2.4 eIQ Tool Installation

The model inference can be done with TensorFlow Lite for Microcontrollers or Glow. If using Glow, You will need to install the latest [Glow package](#). This will put the Glow compiler and helper programs into **C:\NXP\Glow** and add it to your executable path.

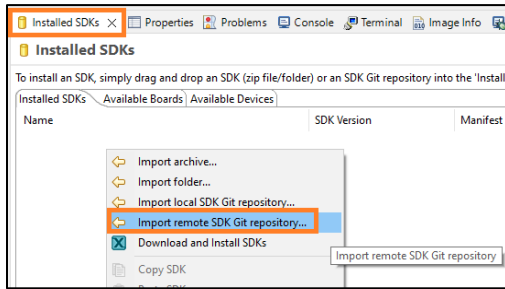
3 Import the App Software Pack into MCUXpresso IDE

There are two methods to get the application software pack. This section will cover both options, but only one needs to be done. It is recommended to use the first option as it is more straight forward.

3.1 Option #1: Get the App Software Pack with MCUXpresso IDE

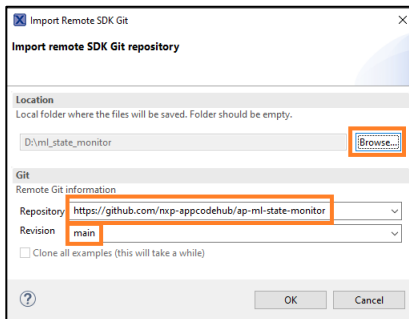
1. Open MCUXpresso IDE and select a workspace location in an empty directory.

- Right click in the blank area of the **Installed SDKs** panel at the bottom and select **Import remote SDK Git repository...**

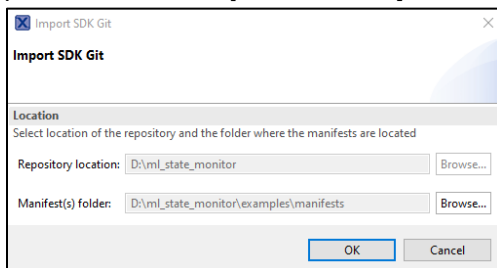


- In the dialog box that comes up:
 - In the **Location** field, click on the Browse button and select or create an empty directory for the application software pack to be downloaded to. Make note of this location as it'll be used throughout this lab.
 - In the **Repository** field put: **https://github.com/nxp-appcodehub/ap-ml-state-monitor**
 - In the **Revision** field put: **main**

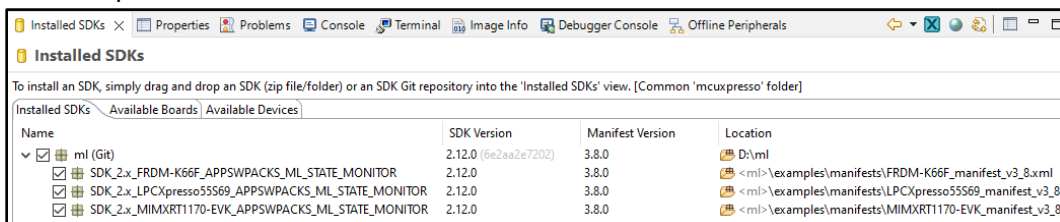
Then hit OK to download the application software pack



- A dialog box will then come up asking for the location of the Manifests folder. Make sure to point it to the **<repo_directory> \examples \manifests** folder if it is not already.



- Once imported the Installed SDKs tab will look like this:



3.2 Option #2: Use the Command Line

1. Open up a Windows command line and execute the following lines:

```
west init -m https://github.com/nxp-appcodehub/ap-ml-state-monitor --mr main appswpacks_ml_state_monitor
cd appswpacks_ml_state_monitor
west update
```

```

D:\ml>west init -m https://github.com/nxp-appcodehub/ap-ml-state-monitor --mr main appswpacks_ml_state_monitor
=== Initializing in D:\ml\appswpacks_ml_state_monitor
--- Cloning manifest repository from https://github.com/nxp-appcodehub/ap-ml-state-monitor , rev. main
Cloning into 'D:\ml\appswpacks_ml_state_monitor\.west\manifest-tmp'...

remote: Enumerating objects: 689, done.
remote: Counting objects: 100% (689/689), done.
remote: Compressing objects: 100% (574/574), done.
remote: Total 1018 (delta 305), reused 302 (delta 106), pack-reused 329
Receiving objects: 100% (1018/1018), 107.68 MiB | 12.54 MiB/s, done.
Resolving deltas: 100% (414/414), done.
Updating files: 100% (503/503), done.
--- setting manifest.path to examples
=== Initialized. Now run "west update" inside D:\ml\appswpacks_ml_state_monitor.

D:\ml>cd appswpacks_ml_state_monitor
D:\ml\appswpacks_ml_state_monitor>west update
=== updating mcux-sdk (core):
--- mcux-sdk: initializing
Initialized empty Git repository in D:\ml\appswpacks_ml_state_monitor\core\.git\
--- mcux-sdk: fetching, need revision MCUX_2.12.0
remote: Enumerating objects: 43491, done.
remote: Counting objects: 100% (43491/43491), done.
remote: Compressing objects: 28% (3211/11466)

```

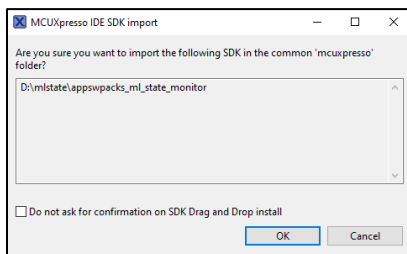
2. Go to the `\appswpacks_ml_state_monitor\examples\manifests` folder and find the `MIMXRT1170-EVK_manifest_v3_8.xml` file. If using a different board copy that board's .xml file instead.

Name	Date modified	Type	Size
FRDM-K66F_manifest_v3_8.xml	9/6/2023 11:01 PM	XML Document	499 KB
LPCXpresso55569_manifest_v3_8.xml	9/6/2023 11:01 PM	XML Document	532 KB
MIMXRT1170-EVK_manifest_v3_8.xml	9/6/2023 11:01 PM	XML Document	615 KB

3. Copy that `MIMXRT1170-EVK_manifest_v3_8.xml` file into the root `\appswpacks_ml_state_monitor\` directory. It should look like the following when done:

Name	Date modified	Type	Size
.west	9/6/2023 11:01 PM	File folder	
core	9/6/2023 11:02 PM	File folder	
examples	9/6/2023 11:01 PM	File folder	
middleware	9/6/2023 11:02 PM	File folder	
rtos	9/6/2023 11:02 PM	File folder	
MIMXRT1170-EVK_manifest_v3_8.xml	9/6/2023 11:01 PM	XML Document	615 KB

6. Next open MCUXpresso IDE and select a workspace location in an empty directory.
7. Drag-and-drop the `\appswpacks_ml_state_monitor` directory that was created in the previous step into the **Installed SDKs** window, located on a tab at the bottom of the screen named "Installed SDKs". You will get the following pop-up, so hit OK.



8. Once imported, the Installed SDK panel will look something like this

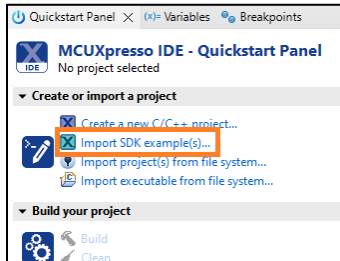
Name	SDK Version	Manifest Version	Location
SDK_2x_MIMXRT1170-EVK_APPSWPACKS_ML_STATE_MONITOR	2.12.0	3.8.0	<Common>\appswpacks_ml_state_monitor

4 Gather the Data

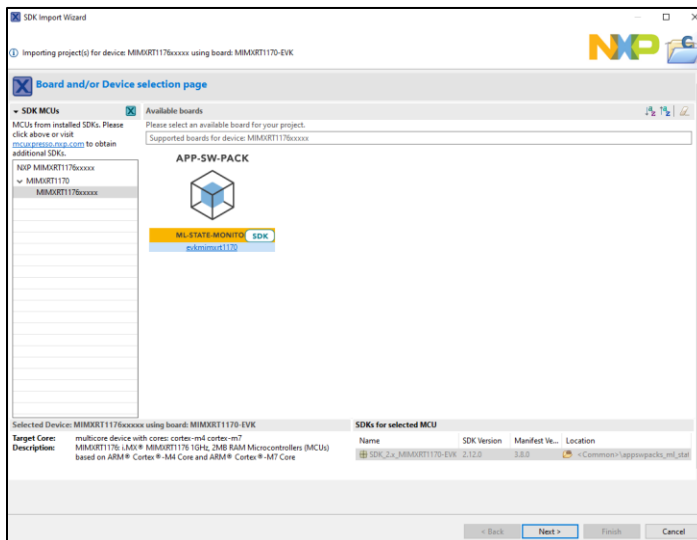
To train a model, sensor data must be collected. For this lab the on-board accelerometer will be used and the data that is collected will be stored on a microSD card.

4.1 Import the ML-based System State Monitor project

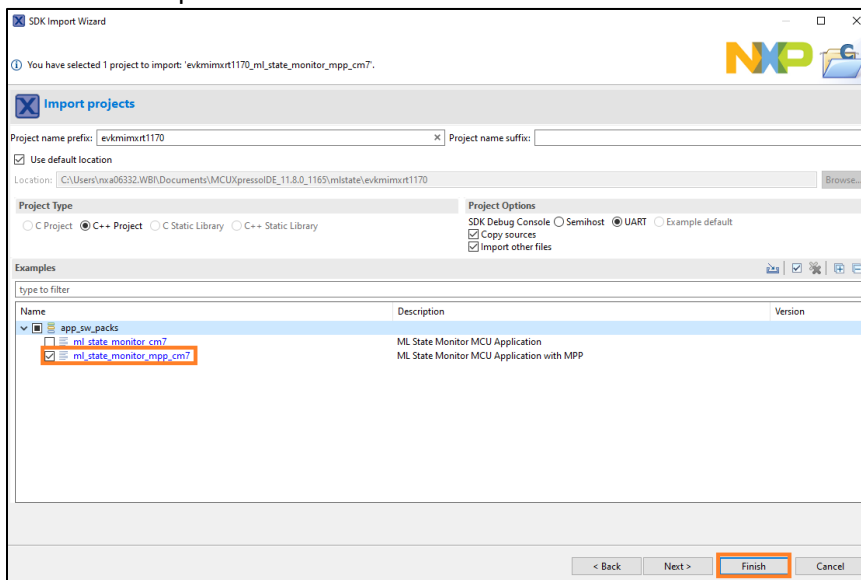
1. In the Quickstart Panel in the lower left corner, click on **Import SDK examples(s)...**



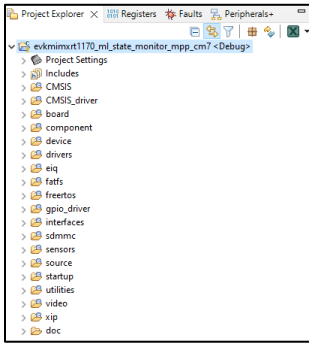
2. Select the **APP-SW-PACK ML-State-Monitor** for evkmimxrt1170 and click on **Next**



3. Expand the **app_sw_packs** category and select the **ml_state_monitor_mpp_cm7** example which is compatible with the FRDM-STBI-A8974 board. Then click on **Finish**



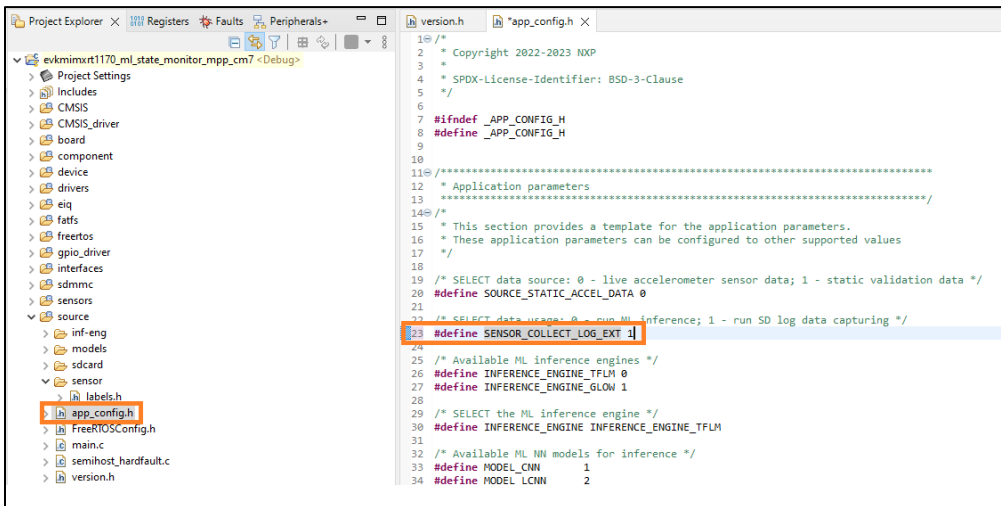
4. Once imported it should look like the following:



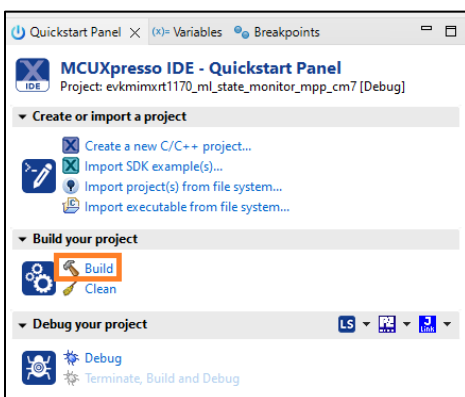
4.2 Collect data

The project will need to be modified to collect accelerometer data and store it on an SD card.

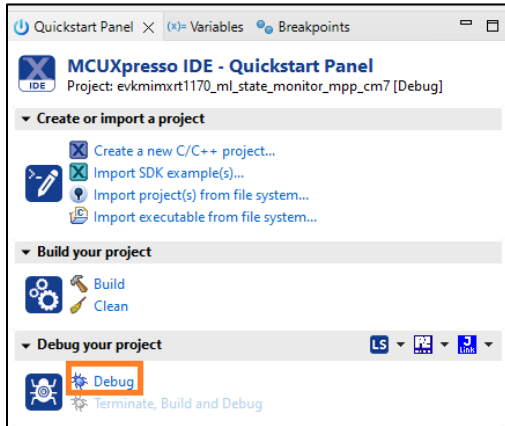
1. Put in a microSD card into the board and set the fan on top of the board. Also make sure nothing is in the M.2 slot in the board as it shares data lines with the SD card.
2. In MCUXpresso IDE, open the **app_config.h** file and on line 23 change the #define for **SENSOR_COLLECT_LOG** to **1** as shown in the image below. This will put the firmware into data capture mode, so that the vibrations generated by the fan or motor will be stored on an SD card.



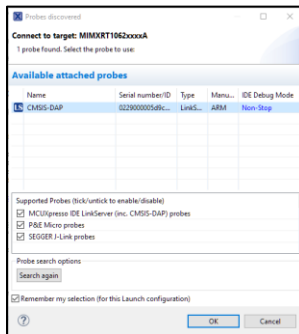
3. Next build the project by clicking on “Build” in the Quickstart Panel and make sure there are no errors.



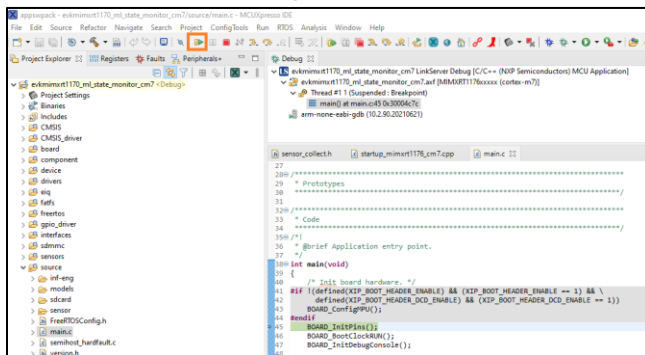
4. Plug the micro-B USB cable into the board at J11 on the i.MXRT1170 board.
5. Open TeraTerm or other terminal program, and connect to the COM port that the board enumerated as. Use 115200 baud, 1 stop bit, no parity.
6. Debug the project by clicking on “Debug” in the Quickstart Panel.



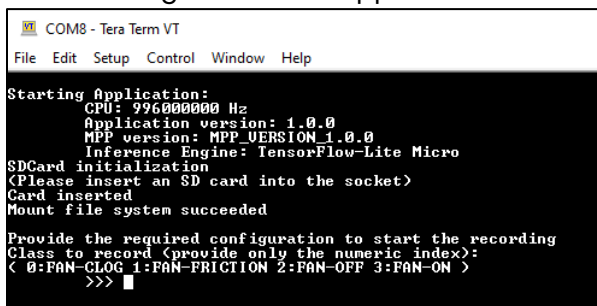
7. It will ask what interface to use. Select CMSIS-DAP.



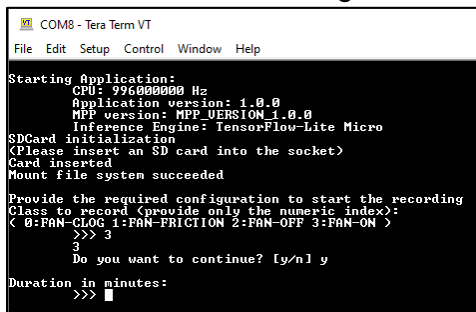
8. The debugger will download the firmware and open up the debug view. Click on the Resume button to start running.



9. The following text should appear in the terminal program:



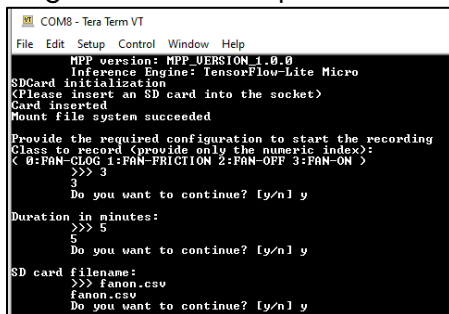
10. For this demo there are four fan states that are going to be detected:
 - FAN-OFF
 - FAN-ON (normal operation)
 - FAN-CLOG (something covering the fan so it does not get the normal airflow)
 - FAN-FRICTION (something creating drag on the blades like a piece of tape or cardboard)
11. Physically put the fan in the desired state. For this first run, lets put the fan into the normal ON state.
12. In the terminal, type of the number of the fan state that corresponds with what the physical fan state is. So for this example, type '3' for FAN-ON.
13. The next selection is how many minutes to gather data. There is no hard and fast rule on how much data needs to be collected, but at a minimum several minutes worth will be required. For this simple example we'll select 5 minutes, but for more complex examples it could be significantly longer to ensure good training results. The [ML-based System State Monitor Dataset Creation Guide](#) provides a much deeper exploration into the optimal data collection techniques, but the key goal is to cover every corner case as much as possible and to not use the same data for training and validation.



```

COM8 - Tera Term VT
File Edit Setup Control Window Help
Starting Application:
CPU: 996000000 Hz
Application version: 1.0.0
MPP version: MPP_VERSION_1.0.0
Inference Engine: TensorFlow-Lite Micro
SDCard initialization
(Please insert an SD card into the socket)
Card inserted
Mount file system succeeded
Provide the required configuration to start the recording
Class to record (provide only the numeric index):
< 0:FAN-CLOG 1:FAN-FRICTION 2:FAN-OFF 3:FAN-ON >
>>> 3
>>> 3
Do you want to continue? [y/n] y
Duration in minutes:
>>>
  
```

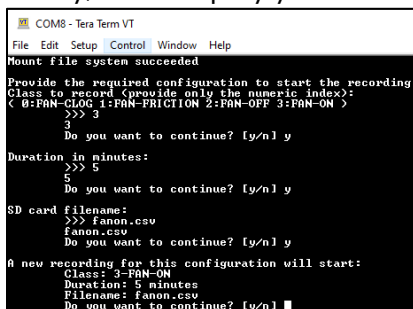
14. Next give a filename where this data will be stored in CSV (Comma Separated Value) format. It can be any name but should be descriptive and cannot be more than 12 characters in length. For this example lets use fanon.csv



```

COM8 - Tera Term VT
File Edit Setup Control Window Help
MPP version: MPP_VERSION_1.0.0
Inference Engine: TensorFlow-Lite Micro
SDCard initialization
(Please insert an SD card into the socket)
Card inserted
Mount file system succeeded
Provide the required configuration to start the recording
Class to record (provide only the numeric index):
< 0:FAN-CLOG 1:FAN-FRICTION 2:FAN-OFF 3:FAN-ON >
>>> 3
>>> 3
Do you want to continue? [y/n] y
Duration in minutes:
>>> 5
Do you want to continue? [y/n] y
SD card filename:
>>> fanon.csv
fanon.csv
Do you want to continue? [y/n] y
  
```

15. Finally, it will display your selections. Type 'y' to then start the data collection:



```

COM8 - Tera Term VT
File Edit Setup Control Window Help
Mount file system succeeded
Provide the required configuration to start the recording
Class to record (provide only the numeric index):
< 0:FAN-CLOG 1:FAN-FRICTION 2:FAN-OFF 3:FAN-ON >
>>> 3
Do you want to continue? [y/n] y
Duration in minutes:
>>> 5
Do you want to continue? [y/n] y
SD card filename:
>>> fanon.csv
fanon.csv
Do you want to continue? [y/n] y
A new recording for this configuration will start:
Class: 3-FAN-ON
Duration: 5 minutes
Filename: fanon.csv
Do you want to continue? [y/n]
  
```

16. You will see the status as it collects the data.
17. Once completed, then you can either remove the SD card to look at the data on your computer or hit the Reset button on the board to start the process again to collect validation data for FAN-ON or to collect the other fan states. You'll want to collect data for all 4 conditions and collect both Training data and Validation data so there should be at least 8 runs. The same amount of data collection time should be used for all data sets.

```

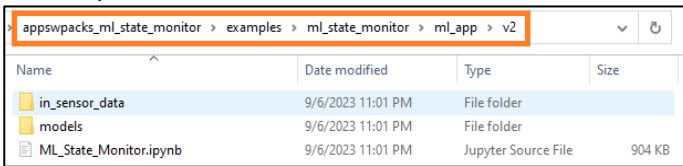
COMS - Tera Term VT
File Edit Setup Control Window Help
Class to record (provide only the numeric index):
< 0:FAN-CLOG 1:FAN-FRICTION 2:FAN-OFF 3:FAN-ON >
>>> 3
3
Do you want to continue? [y/n] y
Duration in minutes:
>>> 5
5
Do you want to continue? [y/n] y
SD card filename:
>>> fanon.csv
fanon.csv
Do you want to continue? [y/n] y
A new recording for this configuration will start:
Class: 3-FAN-ON
Duration: 5 minutes
Filename: fanon.csv
Do you want to continue? [y/n] y
The recording will start in 5 4 3 2 1
Progress: 0.33% (0.02/5 minutes)
  
```

5 Train Data and create a model

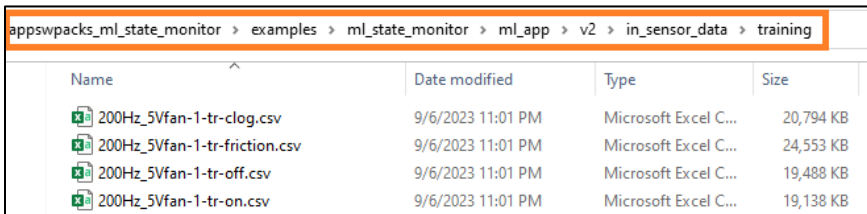
5.1 Copy Training Data to PC

Once all the data required has been saved to the microSD card, remove it from the RT1170 EVK and put it into a card reader for your PC so the data can be copied to the hard drive.

1. Inside the project folder there is a folder named **\appswpacks_ml_state_monitor\examples\ml_state_monitor\ml_app\v2\in_sensor_data**. Locate that directory. Make sure to use the **v2** path.



2. Inside that folder there are two subfolders, one named training and the other validation. By default they contain pre-created data store in .csv files that were gathered from a simple 5V fan.



- These files should be deleted, and the newly collected data placed inside these folders in order to create a new custom model. The file name does not matter but should end in .csv. The files should cover all the possible classes and should be updated for both the **training** and **validation** folders with their respective training and validation data.

Name	Date modified	Type	Size
FANCL0G.CSV	1/1/2018 12:00 AM	Microsoft Excel C...	1,631 KB
FANFRICT.CSV	1/1/2018 12:00 AM	Microsoft Excel C...	1,629 KB
FANOFF.CSV	1/1/2018 12:00 AM	Microsoft Excel C...	1,627 KB
FANON.CSV	1/1/2018 12:00 AM	Microsoft Excel C...	1,778 KB

If you did not collect new data, you can use the pre-existing .csv files as that has data collected with a simple 5V brushless DC motor.

- The Python scripts that will be used in the next section will assume the files will be in the following Comma Separated Value format, which is how the data was written to the files by using the projects from the previous section, so no modifications should be needed.

```

1 class, time [ms], Ax, Ay, Az
2 FAN-CLOG, 5, -134, 69, 994
3 FAN-CLOG, 10, -20, -66, 1077
4 FAN-CLOG, 15, 57, -27, 1213
5 FAN-CLOG, 20, 3, -54, 1092
6 FAN-CLOG, 25, -111, 57, 1107
7 FAN-CLOG, 30, -35, -12, 915
8 FAN-CLOG, 35, 106, 30, 1032
9 FAN-CLOG, 40, -8, 34, 1024
10 FAN-CLOG, 45, -69, -123, 843
11 FAN-CLOG, 50, 19, 3, 990
12 FAN-CLOG, 55, 122, 115, 1126
13 FAN-CLOG, 60, -100, 107, 1043

```

5.2 Train The Model

With the data ready, now use the Jupyter notebook to train the model.

- Open a Windows command prompt
- Navigate to the `\appswpacks_ml_state_monitor\examples\ml_state_monitor\ml_app\v2\` directory
- Start the notebook by typing “**jupyter-notebook**” into the command prompt

```

D:\mlstate\appswpacks_ml_state_monitor\examples\ml_state_monitor\ml_app\v2>jupyter-notebook

Update

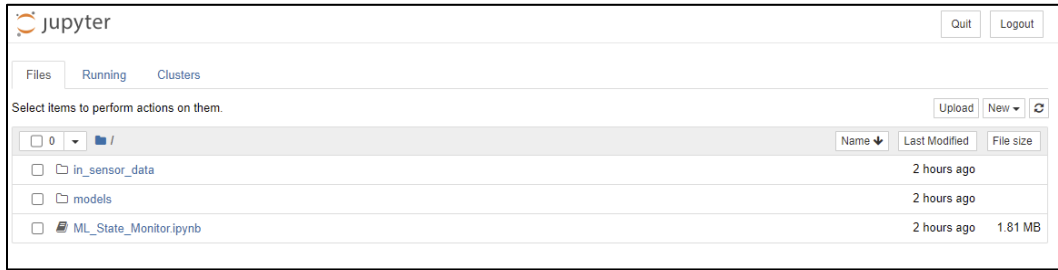
Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extension 5.
https://jupyter-notebook.readthedocs.io/en/latest/migrate_to_notebook7.html
Please note that updating to Notebook 7 might break some of your extensions.

[I 00:06:32.861 NotebookApp] Serving notebooks from local directory: D:\mlstate\appswpacks_ml_state_monitor\examples\ml_state_monitor\ml_app\v2
[I 00:06:32.861 NotebookApp] Jupyter Notebook 6.5.4 is running at:
http://localhost:8888/?token=5b6861deb05e91835d9bf2fiad098d73efb162873ea09f13
or http://127.0.0.1:8888/?token=5b6861deb05e91835d9bf2fiad098d73efb162873ea09f13
Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).

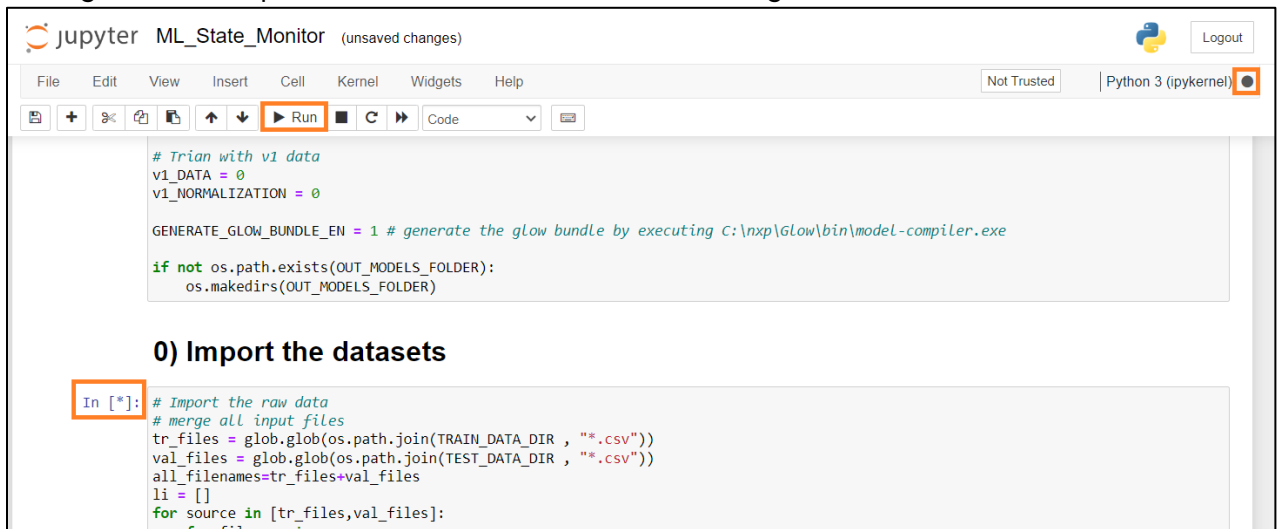
To access the notebook, open this file in a browser:
File:///C:/Users/nx09332.NBI/AppData/Roaming/Jupyter/runtime/nbsrvr-23152-open.html
Or copy and paste one of these URLs:
http://localhost:8888/?token=5b6861deb05e91835d9bf2fiad098d73efb162873ea09f13
or http://127.0.0.1:8888/?token=5b6861deb05e91835d9bf2fiad098d73efb162873ea09f13
0.00s - Debugger warning: It seems that frozen modules are being used, which may
0.00s - make the debugger miss breakpoints. Please pass -Xfrozen_modules-off
0.00s - to python to disable frozen modules.
0.00s - Note: Debugging will proceed. Set PYDEVD_DISABLE_FILE_VALIDATION=1 to disable this validation.

```

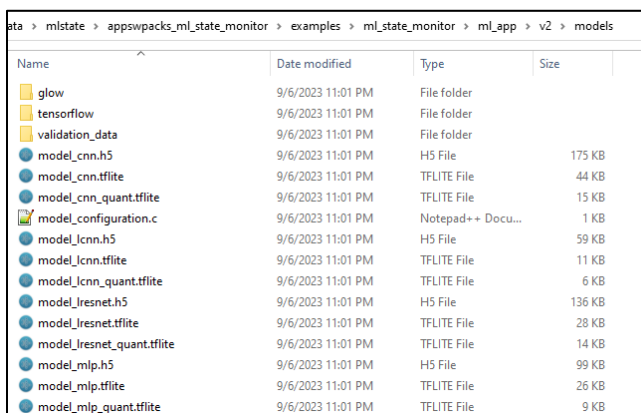
- A web browser window should automatically open up and look like the following. If it does not, there are links in the terminal window that can be copy and pasted into a browser.



- Click on **ML_State_Monitor.ipynb** to open the notebook
- It will look like the following. Press the **Run** button to execute each section of the script. While executing an asterisk (*) will appear and the dot in the upper right will be dark. Don't press Run again until the previous section has finished executing.



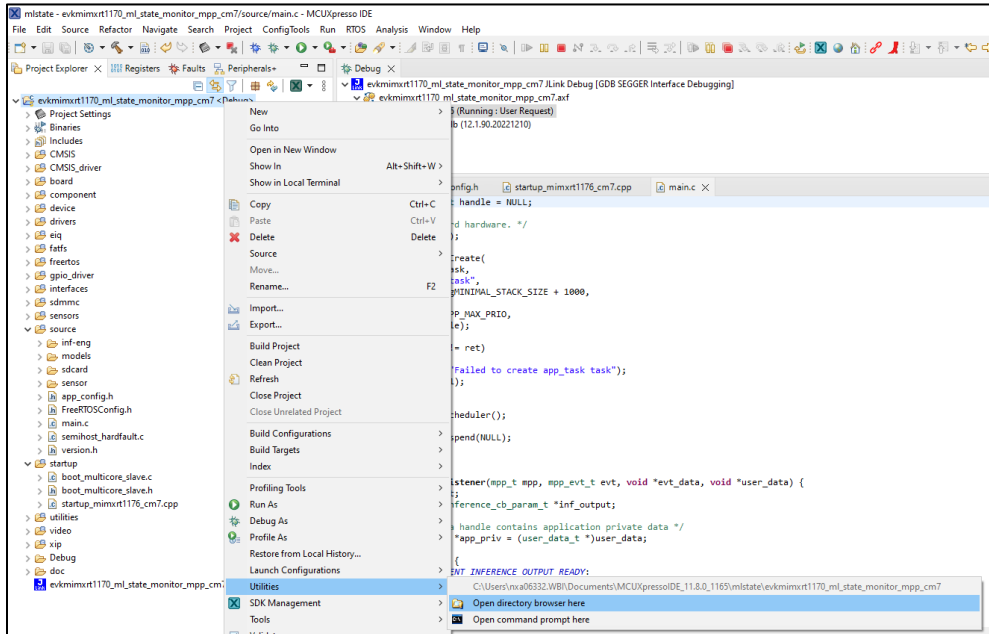
- You may see some warning messages when executing certain blocks of the script but you should not get any error/failure messages.
- The end of the script will export the models to **\appswpacks_ml_state_monitor\examples\ml_state_monitor\ml_app\v2\models**. There will also be C array versions of those models in the **tensorflow** directory and Glow compiled versions of those models in the **glow** directory. Those files will be used in the next section.



6 Update Model for Inference Engines

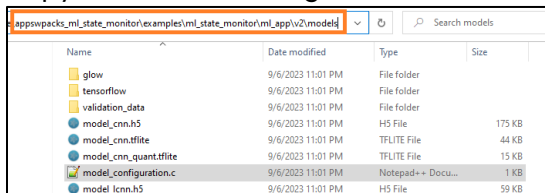
It is now possible to use all 3 eIQ inference engine options for microcontrollers (TFLM, Glow, and DeepViewRT) to run this newly created model on the i.MX RT1170 EVK.

1. First find the location on your PC where the ML-based system state monitor application resides after it was imported into MCUXpresso IDE in the previous section. So inside MCUXpresso IDE, right click on the **evkmimxrt1170_ml_state_monitor_cm7** project name and go down to **Utilities** and select **Open directory browser here** to open a Windows Explorer window at the location.

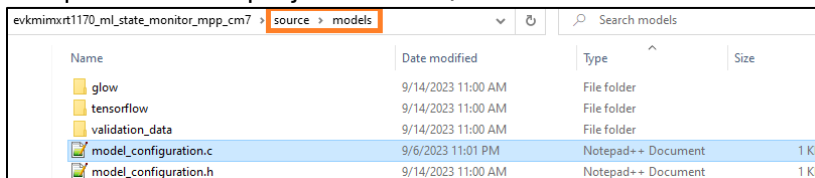


6.1 Update Model Configuration File

1. Some pre-processing parameters for your data was generated during the training phase and put into a file at **\examples\ml_state_monitor\ml_app\v2\model\model_configuration.c**. This new file needs to replace the default model_configuration.c file in the project under **\source\models**.
2. Copy the model_configuration.c file from:



And put it into the project source\models folder:



6.2 TensorFlow Lite for Microcontrollers

3. The Jupyter notebook already has created the C files needed by the TFLM inference engine, so those newly generated files just need to replace the default files in the project
4. Navigate to the <MCUXpresso IDE Workspace>/**source/models/tensorflow** directory.
5. Copy the generated .h and .cpp files from the previous section that were placed in the **\examples\ml_state_monitor\ml_app\models\tensorflow** folder and paste them into this directory.

Copy from:

Path: mlstate > appswpacks_ml_state_monitor > examples > ml_state_monitor > ml_app > v2 > models > tensorflow

Name	Date modified	Type	Size
tensorflow_model_cnn.cpp	9/6/2023 11:01 PM	Notepad++ Docu...	279 KB
tensorflow_model_cnn.h	9/6/2023 11:01 PM	Notepad++ Docu...	1 KB
tensorflow_model_cnn_quant.cpp	9/6/2023 11:01 PM	Notepad++ Docu...	97 KB
tensorflow_model_cnn_quant.h	9/6/2023 11:01 PM	Notepad++ Docu...	1 KB
tensorflow_model_lcn.cpp	9/6/2023 11:01 PM	Notepad++ Docu...	69 KB
tensorflow_model_lcn.h	9/6/2023 11:01 PM	Notepad++ Docu...	1 KB
tensorflow_model_lcn_quant.cpp	9/6/2023 11:01 PM	Notepad++ Docu...	36 KB
tensorflow_model_lcn_quant.h	9/6/2023 11:01 PM	Notepad++ Docu...	1 KB
tensorflow_model_lresnet.cpp	9/6/2023 11:01 PM	Notepad++ Docu...	179 KB
tensorflow_model_lresnet.h	9/6/2023 11:01 PM	Notepad++ Docu...	1 KB
tensorflow_model_lresnet_quant.cpp	9/6/2023 11:01 PM	Notepad++ Docu...	85 KB
tensorflow_model_lresnet_quant.h	9/6/2023 11:01 PM	Notepad++ Docu...	1 KB
tensorflow_model_mlp.cpp	9/6/2023 11:01 PM	Notepad++ Docu...	167 KB
tensorflow_model_mlp.h	9/6/2023 11:01 PM	Notepad++ Docu...	1 KB
tensorflow_model_mlp_quant.cpp	9/6/2023 11:01 PM	Notepad++ Docu...	54 KB
tensorflow_model_mlp_quant.h	9/6/2023 11:01 PM	Notepad++ Docu...	1 KB

Paste into:

Path: mlstate > levkmixrt1170_ml_state_monitor_mpp_cm7 > source > models > tensorflow

Name	Date modified	Type	Size
tensorflow_model_cnn.cpp	9/6/2023 11:52 PM	Notepad++ Document	279 KB
tensorflow_model_cnn.h	9/6/2023 11:52 PM	Notepad++ Document	1 KB
tensorflow_model_cnn_quant.cpp	9/6/2023 11:52 PM	Notepad++ Document	97 KB
tensorflow_model_cnn_quant.h	9/6/2023 11:52 PM	Notepad++ Document	1 KB
tensorflow_model_lcn.cpp	9/6/2023 11:52 PM	Notepad++ Document	69 KB
tensorflow_model_lcn.h	9/6/2023 11:52 PM	Notepad++ Document	1 KB
tensorflow_model_lcn_quant.cpp	9/6/2023 11:52 PM	Notepad++ Document	36 KB
tensorflow_model_lcn_quant.h	9/6/2023 11:52 PM	Notepad++ Document	1 KB
tensorflow_model_lresnet.cpp	9/6/2023 11:52 PM	Notepad++ Document	179 KB
tensorflow_model_lresnet.h	9/6/2023 11:52 PM	Notepad++ Document	1 KB
tensorflow_model_lresnet_quant.cpp	9/6/2023 11:52 PM	Notepad++ Document	85 KB
tensorflow_model_lresnet_quant.h	9/6/2023 11:52 PM	Notepad++ Document	1 KB
tensorflow_model_mlp.cpp	9/6/2023 11:52 PM	Notepad++ Document	167 KB
tensorflow_model_mlp.h	9/6/2023 11:52 PM	Notepad++ Document	1 KB
tensorflow_model_mlp_quant.cpp	9/6/2023 11:52 PM	Notepad++ Document	54 KB
tensorflow_model_mlp_quant.h	9/6/2023 11:52 PM	Notepad++ Document	1 KB

6.3 Glow

1. The Jupyter notebook already has created the Glow bundled needed by the Glow inference engine, so those newly generated files just need to replace the default files in the project. For details on the exact commands used to generate the Glow bundle see cell 29 in that Jupyter notebook.
2. Navigate to the <MCUXpresso IDE Workspace>/**source/models/glow** directory.

- Copy the Glow bundle files from the previous section that were placed in the `\examples \ml_state_monitor \ml_app \v2 \models \glow \cortex_m7` folder and paste them into this directory. Note that you should select the directory in that folder that corresponds with the core architecture in the device that will be used.

Copy from:

appswpacks_ml_state_monitor > examples > ml_state_monitor > ml_app > v2 > models > glow > cortex_m7

Name	Date modified	Type	Size
model_cnn.h	9/6/2023 11:01 PM	Notepad++ Docu...	3 KB
model_cnn.o	9/6/2023 11:01 PM	O File	472 KB
model_cnn.weights.bin	9/6/2023 11:01 PM	BIN File	41 KB
model_cnn.weights.txt	9/6/2023 11:01 PM	Notepad++ Docu...	248 KB
model_cnn_quant.h	9/6/2023 11:01 PM	Notepad++ Docu...	3 KB
model_cnn_quant.o	9/6/2023 11:01 PM	O File	473 KB
model_cnn_quant.weights.bin	9/6/2023 11:01 PM	BIN File	11 KB
model_cnn_quant.weights.txt	9/6/2023 11:01 PM	Notepad++ Docu...	65 KB
model_lcn.h	9/6/2023 11:01 PM	Notepad++ Docu...	3 KB
model_lcn.o	9/6/2023 11:01 PM	O File	472 KB
model_lcn.weights.bin	9/6/2023 11:01 PM	BIN File	9 KB
model_lcn.weights.txt	9/6/2023 11:01 PM	Notepad++ Docu...	52 KB
model_lcn_quant.h	9/6/2023 11:01 PM	Notepad++ Docu...	3 KB
model_lcn_quant.o	9/6/2023 11:01 PM	O File	473 KB
model_lcn_quant.weights.bin	9/6/2023 11:01 PM	BIN File	3 KB
model_lcn_quant.weights.txt	9/6/2023 11:01 PM	Notepad++ Docu...	15 KB
model_lresnet.h	9/6/2023 11:01 PM	Notepad++ Docu...	3 KB
model_lresnet.o	9/6/2023 11:01 PM	O File	475 KB
model_lresnet.weights.bin	9/6/2023 11:01 PM	BIN File	42 KB
model_lresnet.weights.txt	9/6/2023 11:01 PM	Notepad++ Docu...	256 KB
model_lresnet_quant.h	9/6/2023 11:01 PM	Notepad++ Docu...	3 KB
model_lresnet_quant.o	9/6/2023 11:01 PM	O File	475 KB
model_lresnet_quant.weights.bin	9/6/2023 11:01 PM	BIN File	11 KB
model_lresnet_quant.weights.txt	9/6/2023 11:01 PM	Notepad++ Docu...	66 KB
model_mlp.h	9/6/2023 11:01 PM	Notepad++ Docu...	3 KB
model_mlp.o	9/6/2023 11:01 PM	O File	470 KB
model_mlp.weights.bin	9/6/2023 11:01 PM	BIN File	25 KB
model_mlp.weights.txt	9/6/2023 11:01 PM	Notepad++ Docu...	150 KB
model_mlp_quant.h	9/6/2023 11:01 PM	Notepad++ Docu...	3 KB
model_mlp_quant.o	9/6/2023 11:01 PM	O File	472 KB
model_mlp_quant.weights.bin	9/6/2023 11:01 PM	BIN File	7 KB
model_mlp_quant.weights.txt	9/6/2023 11:01 PM	Notepad++ Docu...	39 KB

Paste into:

MCUXpressoIDE_11.8.0_1165 > mistate > evkmimxrt1170_ml_state_monitor_mpp_cm7 > source > models > glow

Name	Date modified	Type	Size
model_cnn.weights.bin	9/6/2023 11:52 PM	BIN File	41 KB
model_cnn_quant.weights.bin	9/6/2023 11:52 PM	BIN File	11 KB
model_lcn.weights.bin	9/6/2023 11:52 PM	BIN File	9 KB
model_lcn_quant.weights.bin	9/6/2023 11:52 PM	BIN File	3 KB
model_lresnet.weights.bin	9/6/2023 11:52 PM	BIN File	42 KB
model_lresnet_quant.weights.bin	9/6/2023 11:52 PM	BIN File	11 KB
model_mlp.weights.bin	9/6/2023 11:52 PM	BIN File	25 KB
model_mlp_quant.weights.bin	9/6/2023 11:52 PM	BIN File	7 KB
model_cnn.h	9/6/2023 11:52 PM	Notepad++ Document	3 KB
model_cnn.weights.txt	9/6/2023 11:52 PM	Notepad++ Document	248 KB
model_cnn_quant.h	9/6/2023 11:52 PM	Notepad++ Document	3 KB
model_cnn_quant.weights.txt	9/6/2023 11:52 PM	Notepad++ Document	65 KB
model_lcn.h	9/6/2023 11:52 PM	Notepad++ Document	3 KB
model_lcn.weights.txt	9/6/2023 11:52 PM	Notepad++ Document	52 KB
model_lcn_quant.h	9/6/2023 11:52 PM	Notepad++ Document	3 KB
model_lcn_quant.weights.txt	9/6/2023 11:52 PM	Notepad++ Document	15 KB
model_lresnet.h	9/6/2023 11:52 PM	Notepad++ Document	3 KB
model_lresnet.weights.txt	9/6/2023 11:52 PM	Notepad++ Document	256 KB
model_lresnet_quant.h	9/6/2023 11:52 PM	Notepad++ Document	3 KB
model_lresnet_quant.weights.txt	9/6/2023 11:52 PM	Notepad++ Document	66 KB
model_mlp.h	9/6/2023 11:52 PM	Notepad++ Document	3 KB
model_mlp.weights.txt	9/6/2023 11:52 PM	Notepad++ Document	150 KB
model_mlp_quant.h	9/6/2023 11:52 PM	Notepad++ Document	3 KB
model_mlp_quant.weights.txt	9/6/2023 11:52 PM	Notepad++ Document	39 KB
model_cnn.o	9/6/2023 11:52 PM	O File	472 KB
model_cnn_quant.o	9/6/2023 11:52 PM	O File	473 KB
model_lcn.o	9/6/2023 11:52 PM	O File	472 KB
model_lcn_quant.o	9/6/2023 11:52 PM	O File	473 KB
model_lresnet.o	9/6/2023 11:52 PM	O File	475 KB
model_lresnet_quant.o	9/6/2023 11:52 PM	O File	475 KB
model_mlp.o	9/6/2023 11:52 PM	O File	470 KB
model_mlp_quant.o	9/6/2023 11:52 PM	O File	472 KB

7 Deploy Model To Board

Now the models can be executed on the i.MX RT1170 EVK.

1. In MCUXpresso IDE, the **evkmimxrt1170_ml_state_monitor_cm7** project should already be imported into your workspace from the previous data collection section.
2. Open the **app_config.h** file under **source** folder by double clicking on it. The modify line 23 to change the **#define SENSOR_COLLECT_LOG_EXT** to **0** to tell the project to now run the inference of the model

```

1  /*
2  * Copyright 2022-2023 NXP
3  *
4  * SPDX-License-Identifier: BSD-3-Clause
5  */
6
7  #ifndef _APP_CONFIG_H
8  #define _APP_CONFIG_H
9
10
11 /*
12 * Application parameters
13 */
14 /*
15 * This section provides a template for the application parameters.
16 * These application parameters can be configured to other supported values
17 */
18
19 /* SELECT data source: 0 - live accelerometer sensor data; 1 - static validation data */
20 #define SOURCE_STATIC_ACCEL_DATA 0
21
22 /* SELECT data usage: 0 - run ML inference; 1 - run SD log data capturing */
23 #define SENSOR_COLLECT_LOG_EXT 0
24
25 /* Available ML inference engines */
26 #define INFERENCE_ENGINE_TFLM 0
27 #define INFERENCE_ENGINE_GLOW 1
28
29 /* SELECT the ML inference engine */
30 #define INFERENCE_ENGINE INFERENCE_ENGINE_TFLM
31
32 /* Available ML NN models for inference */
33 #define MODEL_CNN 1
34 #define MODEL_LCNN 2
35 #define MODEL_MLP 3
36 #define MODEL_LRESNET 4
37
38 /* SELECT ML NN model for inference */
39 #define SELECTED_MODEL MODEL_CNN
40
41 /* SELECT quantization: 0 - no quantization; 1 - use quantized models */
42 #define MODEL_QUANTIZED 0

```

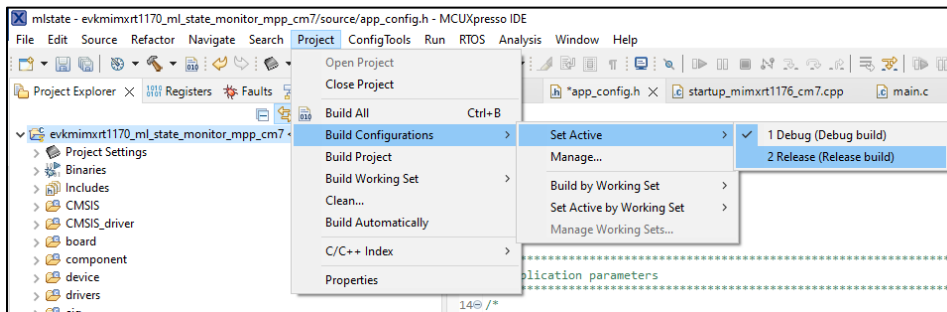
3. There are several other options to modify the inference engine between Glow and TFLM on line 30. By default the project will use TensorFlow Lite for Microcontrollers. It is also possible to select the desired model on line 39. Also a quantized version of the model can be selected on line 42. For the purposes of this lab none of these lines should be changed though.

```

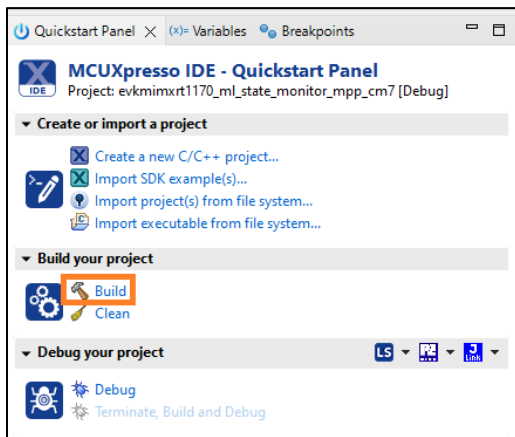
19 /* SELECT data source: 0 - live accelerometer sensor data; 1 - static validation data */
20 #define SOURCE_STATIC_ACCEL_DATA 0
21
22 /* SELECT data usage: 0 - run ML inference; 1 - run SD log data capturing */
23 #define SENSOR_COLLECT_LOG_EXT 0
24
25 /* Available ML inference engines */
26 #define INFERENCE_ENGINE_TFLM 0
27 #define INFERENCE_ENGINE_GLOW 1
28
29 /* SELECT the ML inference engine */
30 #define INFERENCE_ENGINE INFERENCE_ENGINE_TFLM
31
32 /* Available ML NN models for inference */
33 #define MODEL_CNN 1
34 #define MODEL_LCNN 2
35 #define MODEL_MLP 3
36 #define MODEL_LRESNET 4
37
38 /* SELECT ML NN model for inference */
39 #define SELECTED_MODEL MODEL_CNN
40
41 /* SELECT quantization: 0 - no quantization; 1 - use quantized models */
42 #define MODEL_QUANTIZED 0

```

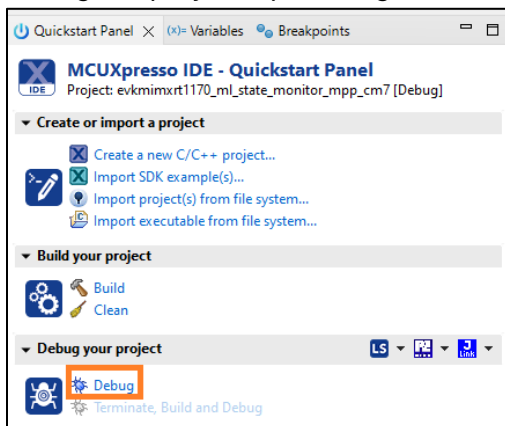
- If using TensorFlow Lite for Microcontrollers, make sure to set the Build Configuration to the “Release” target which does a high compiler optimization. This setting has a minor effect when using Glow or DeepViewRT but significantly reduces inference time when using TFLM. Click on the project name and then go to **Project**→**Build Configurations**→**Set Active**→**Release**



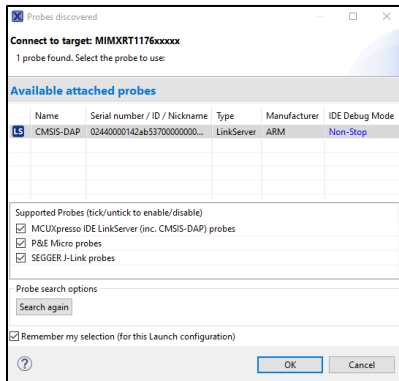
- Next build the project by clicking on “Build” in the Quickstart Panel and make sure there are no errors.



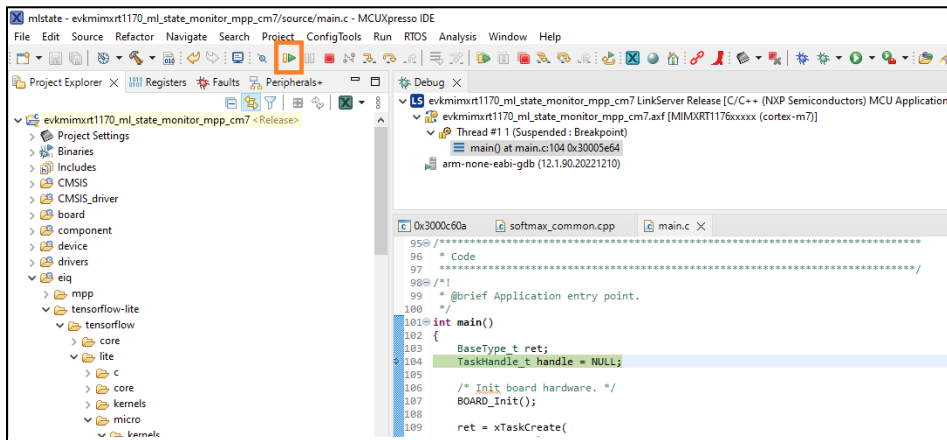
- Plug the micro-B USB cable into the board at J11 on the i.MXRT1170 board.
- Open TeraTerm or other terminal program, and connect to the COM port that the board enumerated as. Use 115200 baud, 1 stop bit, no parity.
- Debug the project by clicking on “Debug” in the Quickstart Panel.



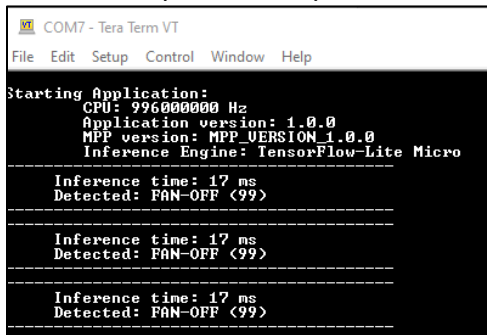
8. It will ask what interface to use. Select CMSIS-DAP.



9. The debugger will download the firmware and open up the debug view. Click on the Resume button to start running.



10. The following text should appear in the terminal program and the current state detected will continue to print out. Try the fan in different states and see it change.



8 Conclusion

This lab demonstrated how to gather and train sensor data to a classification model and deploy it on an i.MX RT device.

This same lab can be used for other types of sensor data models that detect vibrations. By enabling machine learning in embedded systems, there's a wide world of opportunity for new smarter applications.