

Embedded Machine Learning Implementation Using eiQ Time Series on the i.MX RT1052 Kit with Sensor Data

1. Objective

This report aims to provide a technical and instructional guide for implementing embedded machine learning models using the **eiQ Time Series** tool developed by NXP. The practical application is carried out on the **i.MX RT1052** development kit, with input data collected from two sensor sources:

- **MPU6050 accelerometer** (external), configured for vibration analysis.
- **Onboard microphone**, configured for acoustic signal capture.

The focus is on creating and deploying classification or anomaly detection models, using real-world data processed and executed directly on the microcontroller.

2. Methodology

The complete development workflow for implementing embedded machine learning models using sensor data is described below:

2.1 Process Steps

- **Data Logging:**
The firmware running on the i.MX RT1052 kit is capable of capturing two types of signals:
 - **Accelerometer Data:** Acquired from the external **MPU6050** sensor ($\pm 4g$ range), sampled at **1 kHz**, and stored as **.csv** files containing sequential X, Y, Z values per line.
 - **Microphone Data:** Acquired using the onboard microphone, sampled at **16 kHz / 16-bit resolution**, and saved in both **.wav** and **.csv** formats for compatibility

with eiQ Time Series.

- **Data Preprocessing:**

Files are preprocessed using the **Data Operations** section within eiQ Time Series. Operations may include normalization, filtering, and sample alignment.

For accelerometer signals:

- According to the Nyquist Theorem, only frequency components **below 500 Hz** should be analyzed (sampling rate = 1 kHz).
- The $\pm 4g$ range is suited for light to moderate vibrations. For high-impact detection, a wider range ($\pm 8g$ or $\pm 16g$) should be configured.

- For microphone signals:

- The signal is sampled at 16 kHz, allowing analysis of frequencies up to 8 kHz.
- These recordings are well-suited for audio classification, acoustic anomaly detection, or condition monitoring.

- Signal acquisition window size (e.g., 256 samples per line for accelerometer data) and duration (5 seconds) can be adjusted in firmware to better fit the target use case.

- **Dataset Creation:**

Two datasets should be prepared for each application:

- One representing normal signal behavior.
- One representing anomalous or multi-class behavior.

- **Training and Benchmarking:**

The **eiQ Time Series** platform automatically benchmarks multiple algorithms, evaluating performance based on accuracy, model size, and memory footprint to identify the best candidate for deployment.

- **Emulation:**

Trained models are validated using test data not included in the training set. This step helps assess generalization performance.

- **Deployment:**

The final model is exported as a precompiled library (`libtss.a`) and linked to the firmware application on the i.MX RT1052.

3. Configuration

1. Clone the project repository:
https://github.com/jvmreis/nxp_embedded_machine_learning
2. Open **MCUXpresso IDE**.
3. Navigate to **File > Import Project from File System**.
4. Select the cloned project directory (e.g.,
`Github\nxp_embedded_machine_learning`).
5. Configure the project to link the precompiled libtss.a library:
 - a. The `libtss.a` file generated during the **Deployment** phase in eiQ Time Series must be correctly linked into your firmware project. Follow the steps below:
 - b. Right-click on the project in the **Project Explorer**, then select **Properties**.
 - c. Navigate to:
`C/C++ Build > Settings > MCU Linker > Libraries`
 - d. In "**Libraries (-I)**", add: `tss`
 - e. In "**Library search path (-L)**", add:
`$(workspace_loc:/${ProjName}/source/lib)`

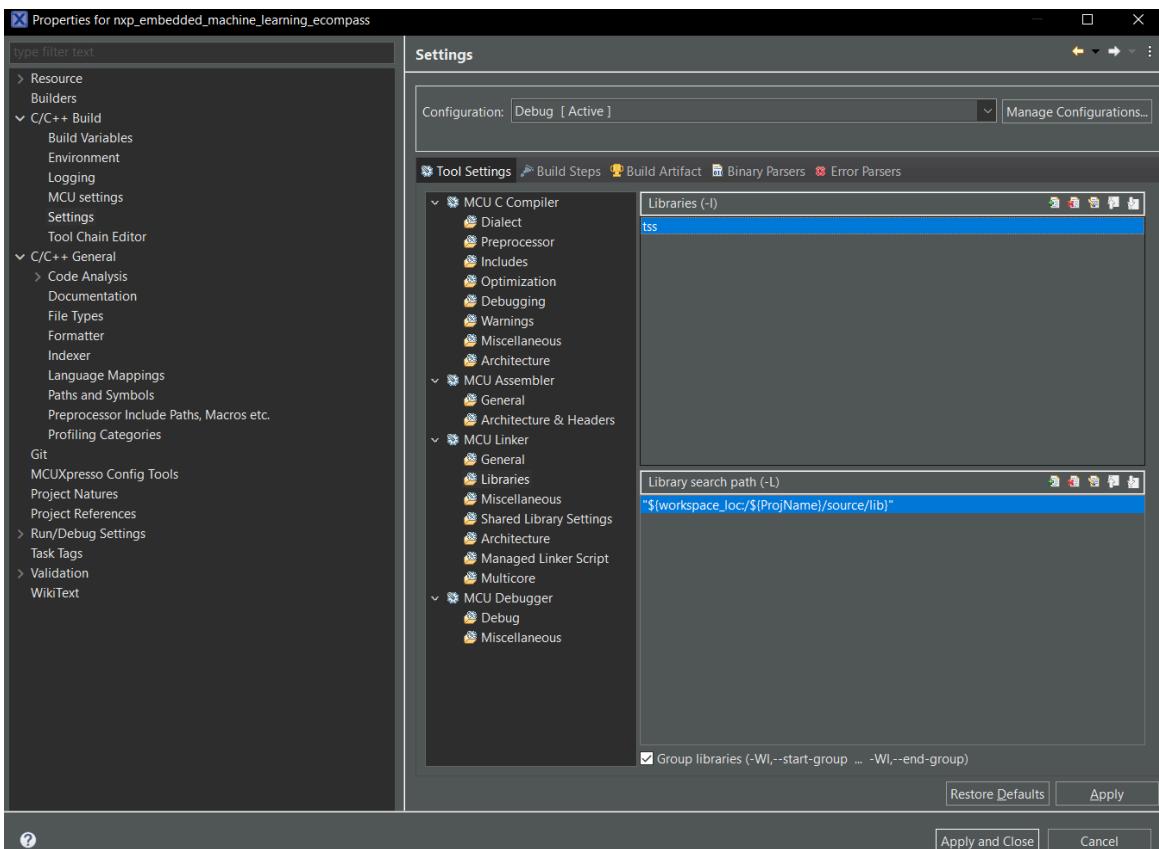


Figure 1 – Linker Configuration in MCUXpresso IDE for the `tss` Library

-
6. Click **Build** to compile the project.

3.1 Tools Used

- **Platform:** NXP i.MX RT1052
- **IDE:** MCUXpresso IDE
- **Serial Terminal:** Tera Term
- **AI Tool:** eiQ Time Series
- **Acceleration Sensor:** MPU6050
- **Storage Medium:** SD Card

This code is using an **external MPU6050** accelerometer, connect it to the development board using the following pin headers. If instead you prefer using the **on-board FXOS8700CQ accelerometer**, you will need to modify the firmware accordingly.

Signal	Pin Header	Pin Number
VCC	J24	Pin 8
GND	J24	Pin 7
INT	J22	Pin 8
SDA	J23	Pin 9
SCL	J23	Pin 8

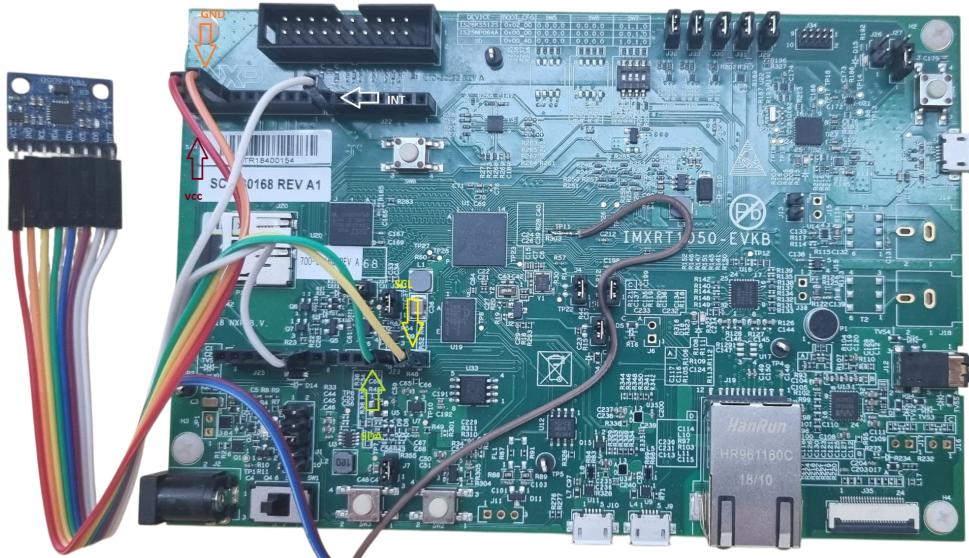


Figure 2 – Pinout Location on i.MX RT1050 EVK Board

4. Practical Experiment Guide

4.1 Data Logging

1. Connect to the i.MX RT1052 board using **Tera Term** at **115200 baud**.
2. Insert an SD card.
3. At startup, the terminal displays the following menu:

A screenshot of the Tera Term VT window. The title bar says "VT COM3 - Tera Term VT". The menu bar includes File, Edit, Setup, Control, Window, and Help. The main window displays a menu:

```
File Edit Setup Control Window Help
Create directory......
Directory exists.

Please choose the option :
1. Anomaly detection embedded Machine Learning Model
2. Classification embedded Machine Learning Model
3. Exclude files
4. Record to SDcard Ext Accelerometer Data
5. Record to SDcard Microphone Data
6. Quit
```

Figure 3 – Menu inference via serial terminal.

- **Option 4:** Captures 5 seconds of acceleration data from MPU6050. Stored as **.csv** (e.g., **ACCELE_1.csv**).
- **Option 5:** Captures 5 seconds of audio data from the onboard microphone. Stored as both **.wav** and **.csv**.

Use both options to collect signals for each class (normal, anomalous).

4.2 Model Training in eiQ Time Series

- Open **eiQ Time Series**.
- Choose the desired model type: **Classification** or **Anomaly Detection**.
- Create a new project with the following settings:
 - Platform:** i.MX RT1050
 - Number of Channels:** 3 (X, Y, Z)
 - RAM/Flash:** Default (or customized, if required)

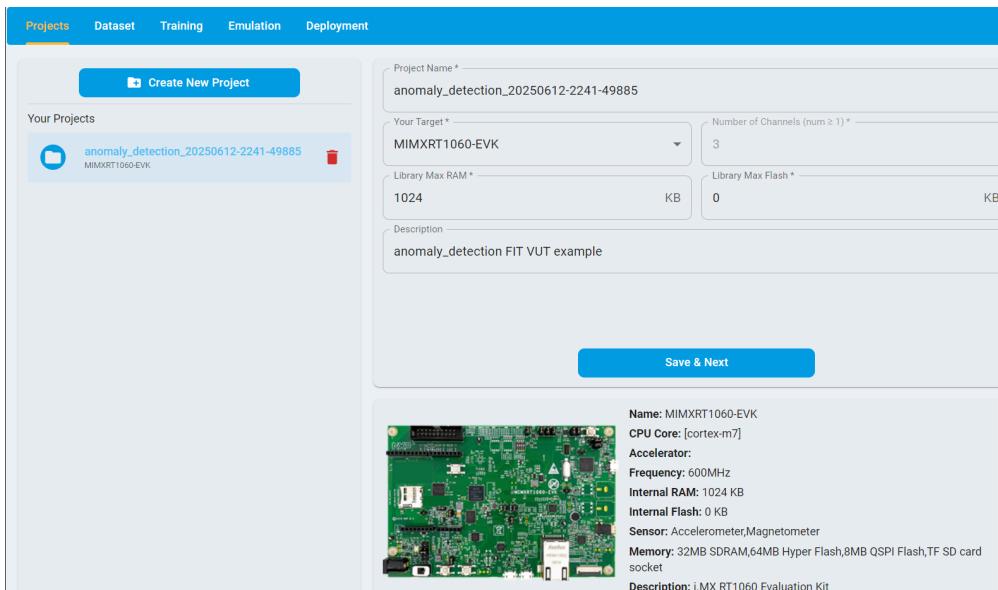


Figure 4 – eiQ project creation screen with model type and hardware settings.

Upload the corresponding **.csv** files with the collected signals.

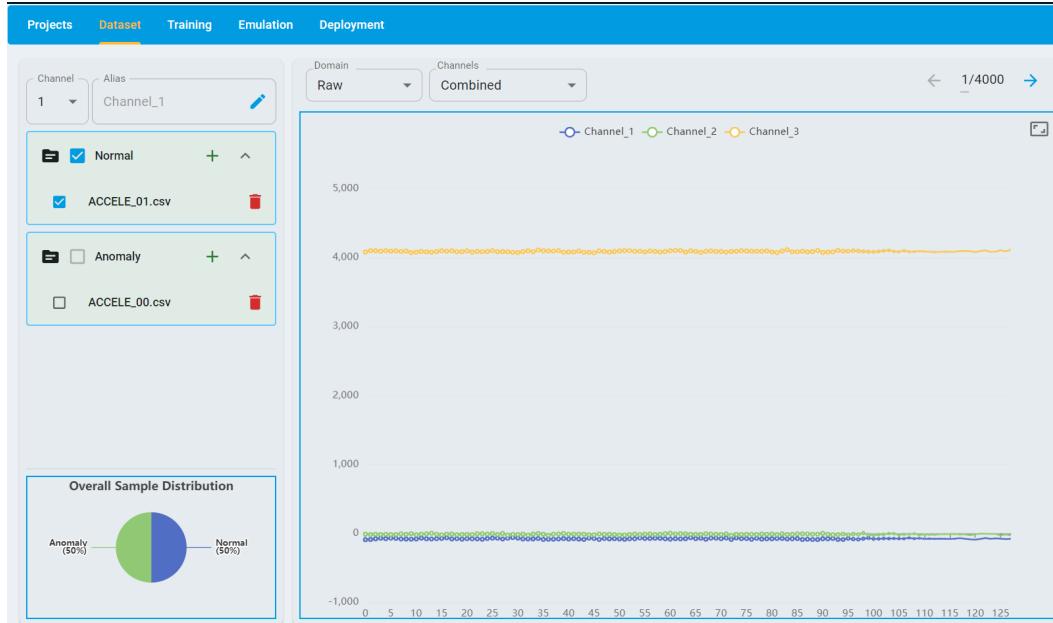


Figure 5 – eiQ project Dataset creation screen with data upload

4.3 Training and Benchmarking

1. Navigate to the **Training** tab.
2. Click **Start New Training**.
3. Wait for the automatic benchmarking process, where different algorithms are tested. The tool will recommend the best-performing model based on accuracy and footprint.

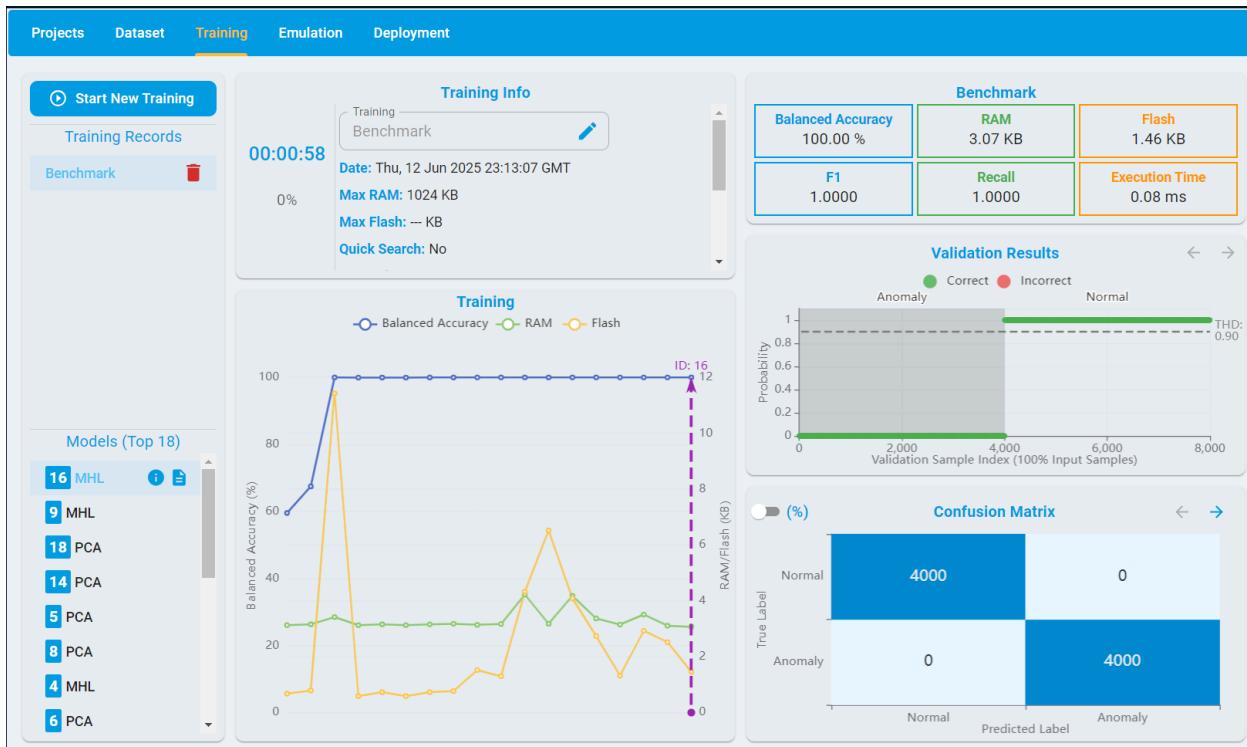


Figure 6 – Automatic benchmarking output and model selection result.

4.4 Emulation

1. Select the trained model.
2. Go to the **Emulation** tab.
3. Upload test signals not used during training.
4. Analyze the model's output based on similarity with known patterns.

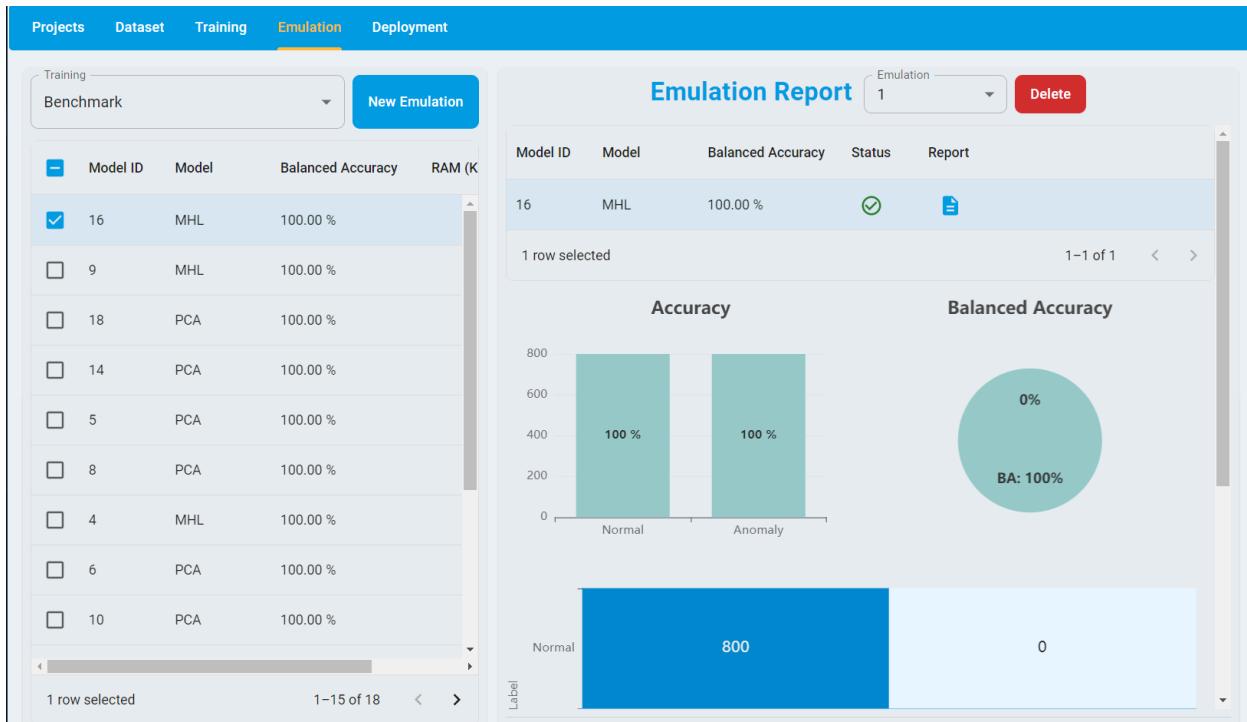


Figure 7 – Emulation results showing signal similarity and inference response.

4.5 Deployment to the Microcontroller

Once the training process is complete in **eiQ Time Series**, the selected model can be exported by clicking **Generate** in the **Deployment** tab. This action generates a precompiled static library (**libtss.a**) that must be integrated into the firmware running on the i.MX RT1052 development board.

4.5.1 Steps to Deploy and Run the Model:

1. Extract the contents of the exported **.zip** package and place the **libtss.a** file into the project directory at:
`C:\Github\nxp_embedded_machine_learning\source\lib`
2. In **MCUXpresso IDE**, open the project settings:
 - o Right-click the project → **Properties**
 - o Go to: **C/C++ Build > Settings > MCU Linker > Libraries**

- Add `tss` in **Libraries (-I)**
 - Add `${workspace_loc}:/${ProjName}/source/lib}` in **Library search path (-L)**
3. Rebuild the project and flash the updated firmware to the i.MX RT1052 using the debugger or USB flasher.
 4. Open **Tera Term** (or another serial terminal) and connect to the board at **115200 baud**. The following menu will appear as the figure 3.
 - **Option 1:** Starts real-time inference using the **anomaly detection model**.
 - **Option 2:** Starts real-time inference using the **classification model**.

4.5.2 Selecting the Sensor Input (Accelerometer or Microphone):

The input source for the machine learning model can be configured in the file `machine_learning.c`.

Inside this file, two core functions are used to collect input data:

- `acc_sample_data(float *data_buffer)` — captures one window of acceleration data (X, Y, Z axes).
- `mic_sample_data(float *data_buffer)` — captures one window of raw audio data (512 samples at 16 kHz, 16-bit).

To change the input source used for inference, simply modify the function call inside either:

- `ml_anomaly_detection()` (uses accelerometer by default)
- `ml_classification()` (uses microphone by default)

For example, to test classification based on accelerometer instead of microphone, you could replace:

```
mic_sample_data(mic_data_input);

status = tss_cls_predict_freqModel(mic_data_input, probabilities,
&class_index);
```

```
acc_sample_data(acc_data_input);

status = tss_cls_predict_freqModel(acc_data_input, probabilities,
&class_index);
```

This allows developers to flexibly switch between sensor modalities depending on the application—whether it's vibration analysis or sound-based detection.

4.5.3 Default configuration

By default, the embedded firmware is pre-configured to demonstrate two machine learning use cases:

- **Anomaly Detection Model (Option 1 in UART menu):**

Uses the **external accelerometer (MPU6050)** to distinguish between two states:

- **Idle sensor** (stationary condition)
- **Shaken sensor** (induced vibration or movement)

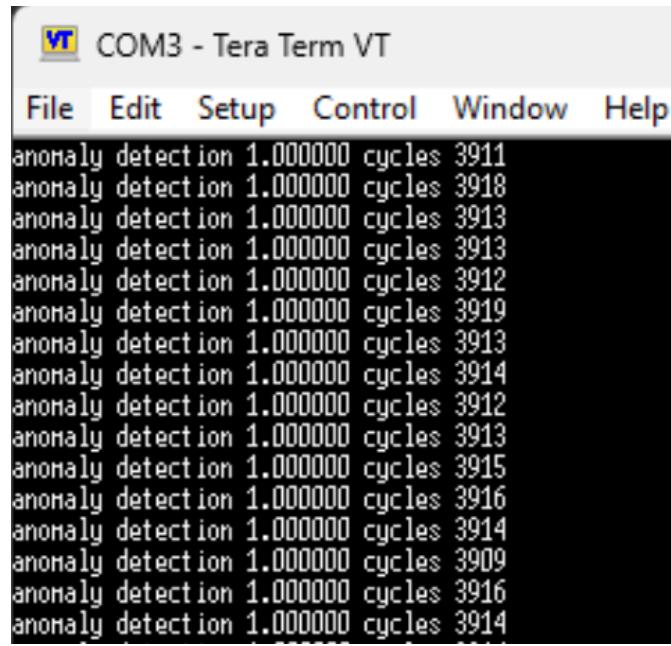
- **Classification Model (Option 2 in UART menu):**

Uses the **onboard microphone** to classify **sine wave sounds** at different frequencies.

The model outputs the following probabilities:

- probabilities[0] → 4000 Hz
- probabilities[1] → 2000 Hz
- probabilities[2] → 1780 Hz
- probabilities[3] → 860 Hz
- probabilities[4] → 244 Hz

These configurations are intended for demonstration and testing purposes and can be customized in the `machine_learning.c` file to suit other sensor data or classification targets.

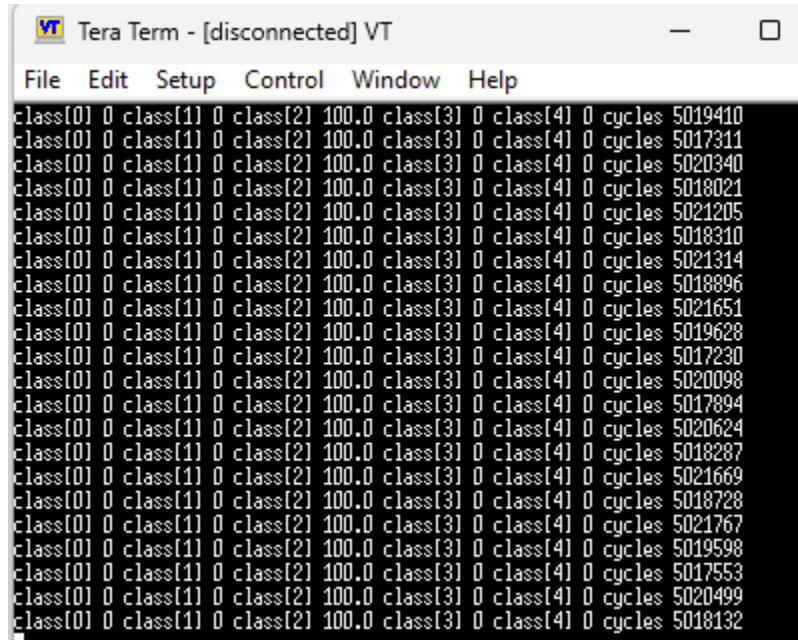


VT COM3 - Tera Term VT

File Edit Setup Control Window Help

```
anomaly detection 1.000000 cycles 3911
anomaly detection 1.000000 cycles 3918
anomaly detection 1.000000 cycles 3913
anomaly detection 1.000000 cycles 3913
anomaly detection 1.000000 cycles 3912
anomaly detection 1.000000 cycles 3919
anomaly detection 1.000000 cycles 3913
anomaly detection 1.000000 cycles 3914
anomaly detection 1.000000 cycles 3912
anomaly detection 1.000000 cycles 3913
anomaly detection 1.000000 cycles 3915
anomaly detection 1.000000 cycles 3916
anomaly detection 1.000000 cycles 3914
anomaly detection 1.000000 cycles 3909
anomaly detection 1.000000 cycles 3916
anomaly detection 1.000000 cycles 3914
```

Figure 8 – Embedded anomaly detection model with accelerometer signal running



VT Tera Term - [disconnected] VT

File Edit Setup Control Window Help

```
class[0] 0 class[1] 0 class[2] 100.0 class[3] 0 class[4] 0 cycles 5019410
class[0] 0 class[1] 0 class[2] 100.0 class[3] 0 class[4] 0 cycles 5017311
class[0] 0 class[1] 0 class[2] 100.0 class[3] 0 class[4] 0 cycles 5020340
class[0] 0 class[1] 0 class[2] 100.0 class[3] 0 class[4] 0 cycles 5018021
class[0] 0 class[1] 0 class[2] 100.0 class[3] 0 class[4] 0 cycles 5021205
class[0] 0 class[1] 0 class[2] 100.0 class[3] 0 class[4] 0 cycles 5018310
class[0] 0 class[1] 0 class[2] 100.0 class[3] 0 class[4] 0 cycles 5021314
class[0] 0 class[1] 0 class[2] 100.0 class[3] 0 class[4] 0 cycles 5018896
class[0] 0 class[1] 0 class[2] 100.0 class[3] 0 class[4] 0 cycles 5021651
class[0] 0 class[1] 0 class[2] 100.0 class[3] 0 class[4] 0 cycles 5019628
class[0] 0 class[1] 0 class[2] 100.0 class[3] 0 class[4] 0 cycles 5017230
class[0] 0 class[1] 0 class[2] 100.0 class[3] 0 class[4] 0 cycles 5020098
class[0] 0 class[1] 0 class[2] 100.0 class[3] 0 class[4] 0 cycles 5017894
class[0] 0 class[1] 0 class[2] 100.0 class[3] 0 class[4] 0 cycles 5020624
class[0] 0 class[1] 0 class[2] 100.0 class[3] 0 class[4] 0 cycles 5018287
class[0] 0 class[1] 0 class[2] 100.0 class[3] 0 class[4] 0 cycles 5021669
class[0] 0 class[1] 0 class[2] 100.0 class[3] 0 class[4] 0 cycles 5018728
class[0] 0 class[1] 0 class[2] 100.0 class[3] 0 class[4] 0 cycles 5021767
class[0] 0 class[1] 0 class[2] 100.0 class[3] 0 class[4] 0 cycles 5019598
class[0] 0 class[1] 0 class[2] 100.0 class[3] 0 class[4] 0 cycles 5017553
class[0] 0 class[1] 0 class[2] 100.0 class[3] 0 class[4] 0 cycles 5020499
class[0] 0 class[1] 0 class[2] 100.0 class[3] 0 class[4] 0 cycles 5018132
```

Figure 9 – Embedded Classification model with microphone signal running

7. Reference

NXP RM – i.MX RT1050 Processor Reference Manual, section 51.5.1.1 (Restart Mode)
<https://www.nxp.com/webapp/Download?colCode=IMXRT1050RM>

Github Repository

https://github.com/jvmreis/nxp_embedded_machine_learning