Embedded Machine Learning Implementation Using eiQ Time Series on the i.MX RT1052 Kit with Acceleration 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 external acceleration data captured by the **MPU6050** sensor. The focus is on creating and deploying classification or anomaly detection models, using real-world data processed and embedded directly on the microcontroller.

---

2. Methodology

The complete development workflow for implementing an embedded *machine learning* model is described below:

2.1 Process Steps

- **Data Logging:**
  The firmware running on the i.MX RT1052 kit is used to capture acceleration signals from the MPU6050 sensor, recording data from the X, Y, and Z axes at a sampling rate of **1 kHz**, over **5-second windows**. The data is stored in `.csv` files on the SD card, with samples arranged sequentially as (X Y Z).

- **Data Preprocessing:**
  The collected data files are processed directly within the **eiQ Time Series** platform, in the *Data Operations* section. Techniques such as **normalization**, **filtering**, and specific adjustments can be applied to ensure data integrity and consistency for training.

  It is essential to understand the signal characteristics and the application domain. The accelerometer is configured with a sampling rate of **1 kHz**, meaning—according to the Nyquist Theorem—that only frequency components **below 500 Hz** should be analyzed to avoid aliasing.

Additionally, the sensor is operating within a **±4g range**, suitable for applications involving **light to moderate vibrations** and **low-impact events**. For use cases with higher intensity events (e.g., collisions, drops), it is recommended to reconfigure the sensor to a wider range, such as **±8g or ±16g**.

Parameters such as the **capture window size** (currently 128 samples per line) and the **total acquisition time** directly affect model performance and can be adjusted based on signal characteristics and anomaly types. All these settings can be modified in the firmware running on the development kit.

- **Dataset Creation:**
  Two datasets should be prepared: one representing the system's normal behavior and another with altered (anomalous or multi-class) signals. These datasets feed the training process of the model.

- **Training and Benchmarking:**
  The eiQ platform automatically benchmarks multiple algorithms, evaluating metrics such as accuracy, model size, and resource usage, to determine the most suitable model for deployment.

- **Emulation:**
  Trained models are validated using unseen data to evaluate the model's generalization capability.

- **Deployment:**
  The selected model is exported as a compiled library (`libtss.a`) and integrated into the application firmware running on the i.MX RT1052 kit.

---

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 (-l)"**, add: `tss`
    e. In **"Library search path (-L)"**, add:
    `${workspace_loc:/${ProjName}/source/lib}`

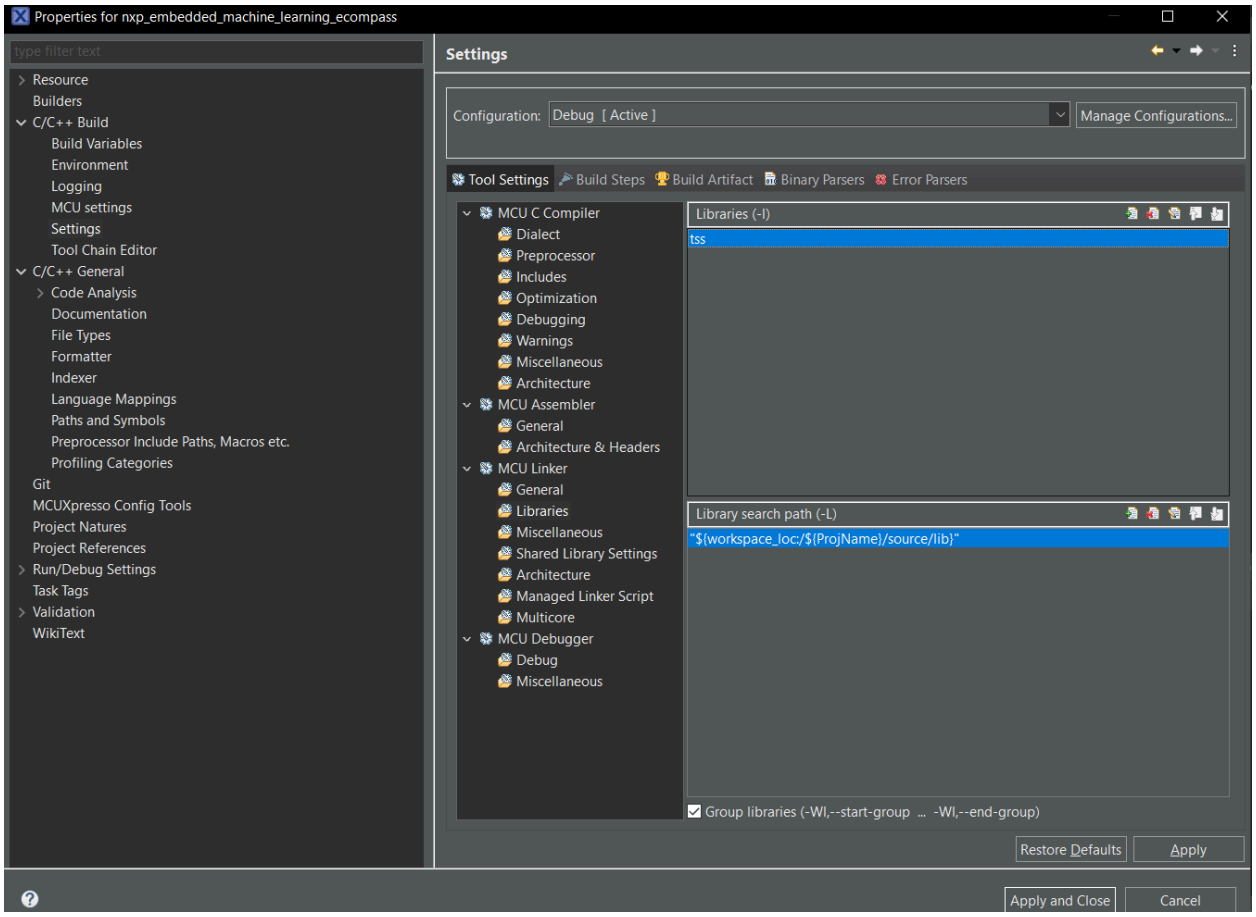

Figure 1 *– Linker Configuration in MCUXpresso IDE for the* `tss` *Library*

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

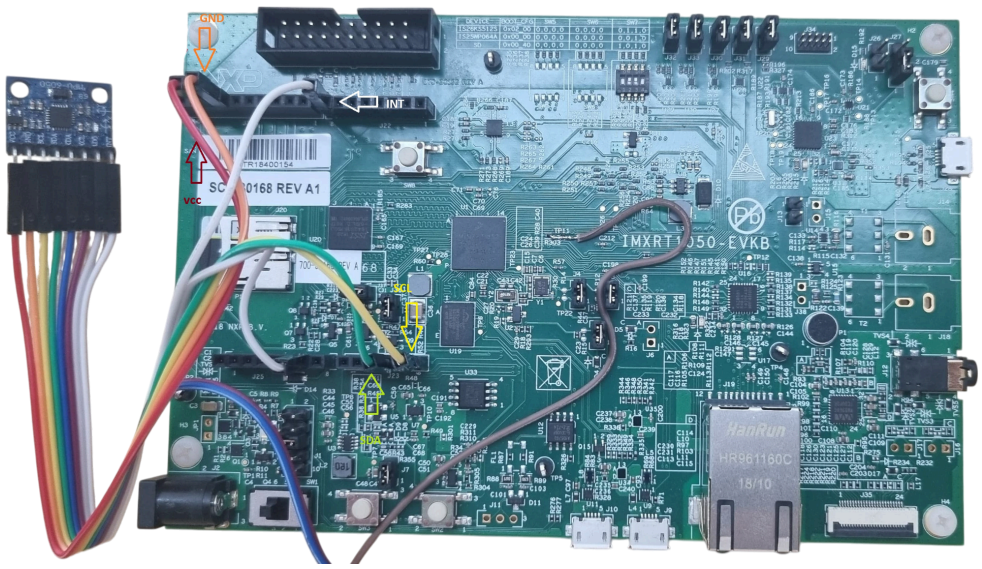

Figure 2 *– Pinout Location on i.MX RT1050 EVK Board*

---

4. Practical Experiment Guide

4.1 Data Logging

1. Connect to the i.MX RT1052 board using a serial terminal (e.g., Tera Term) with a baud rate of **115200**.
2. Insert an SD card into the board.

The pre-loaded firmware is configured to capture **5 seconds of acceleration data at 1 kHz**, using the MPU6050 sensor set to a **±4g range**. The output is saved in `.csv` format and ready to be used in the eiQ platform.

Upon startup, the terminal will display a menu with three options:

- **Option 2:** Delete previously saved files from the SD card to avoid conflicts.
- **Option 3:** Start a new 5-second data acquisition session.

Each new session generates a file named `ACCELE_x.csv`, where `x` is an incremental identifier. You should collect signals representing both **normal** and **anomalous** system behavior.

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:
    i. **Platform:** i.MX RT1050
    ii. **Number of Channels:** 3 (X, Y, Z)
    iii. **RAM/Flash:** Default (or customized, if required)

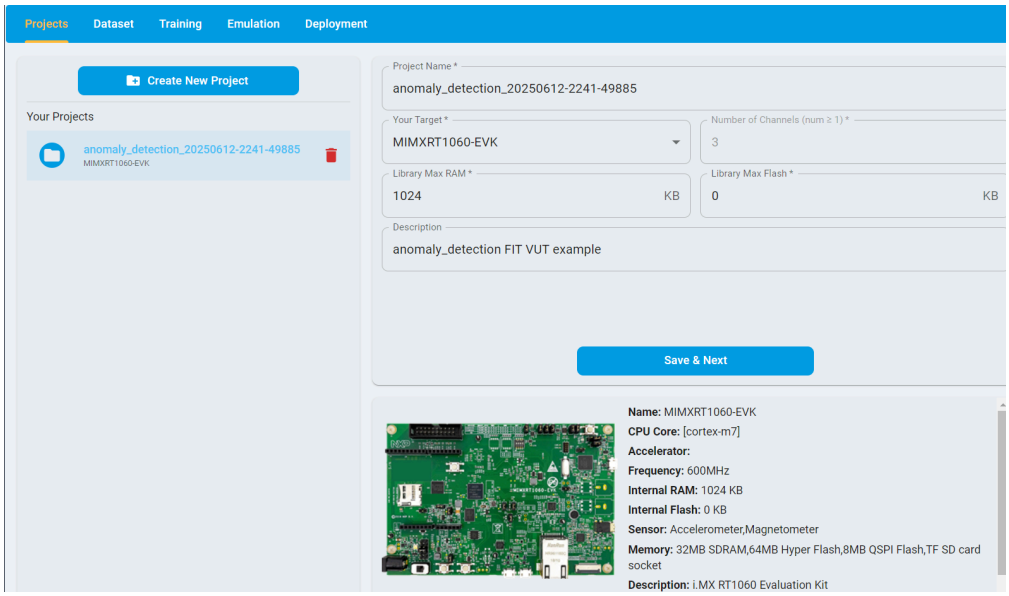

Figure 3 – *eiQ project creation screen with model type and hardware settings.*

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

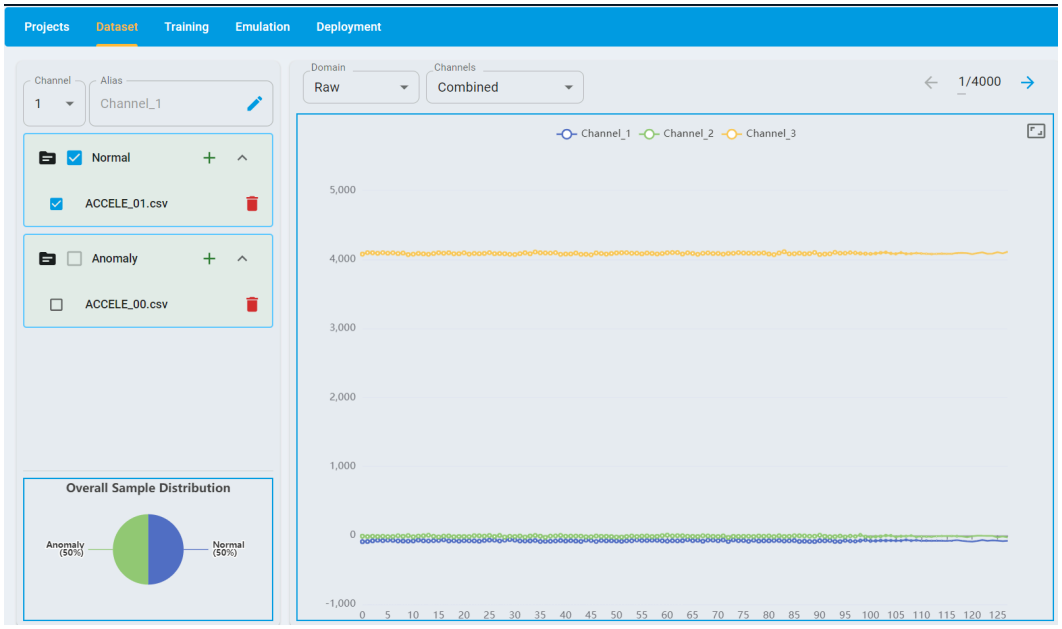

Figure 3 – *eiQ project Dataset creation screen with data upload*

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.

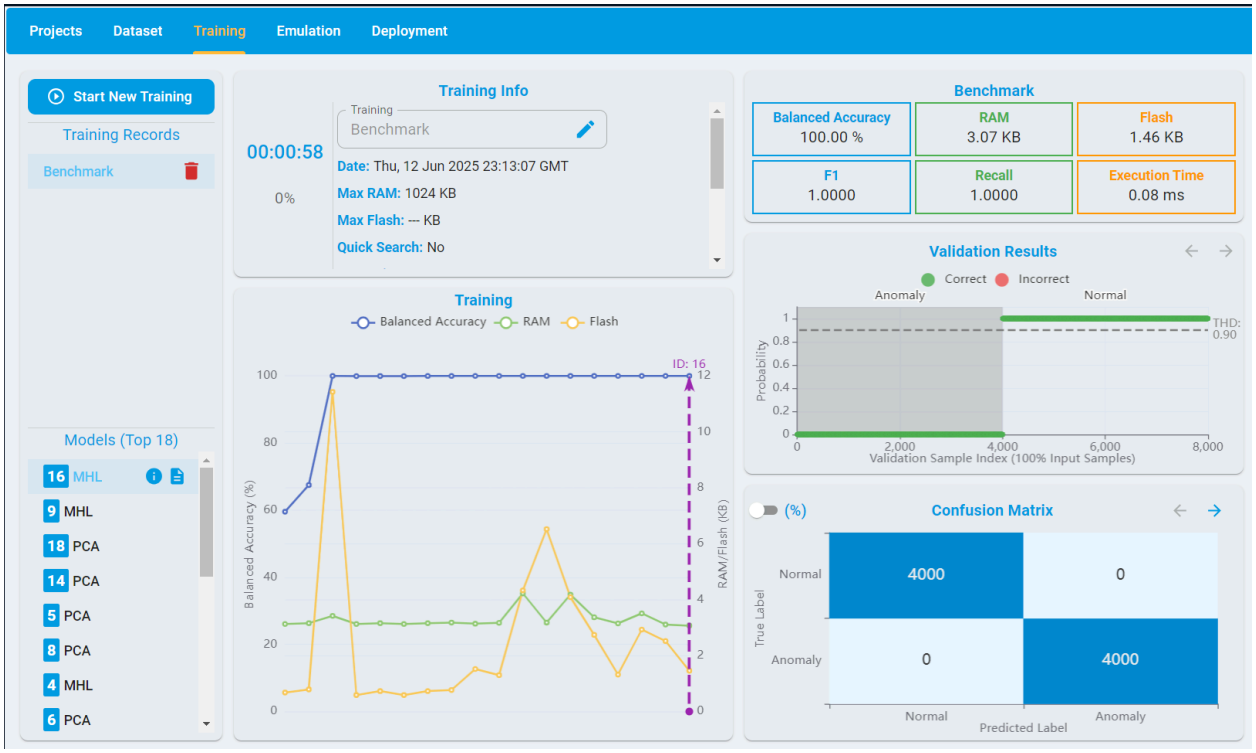

Figure 5 – *Automatic benchmarking output and model selection result.*

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.

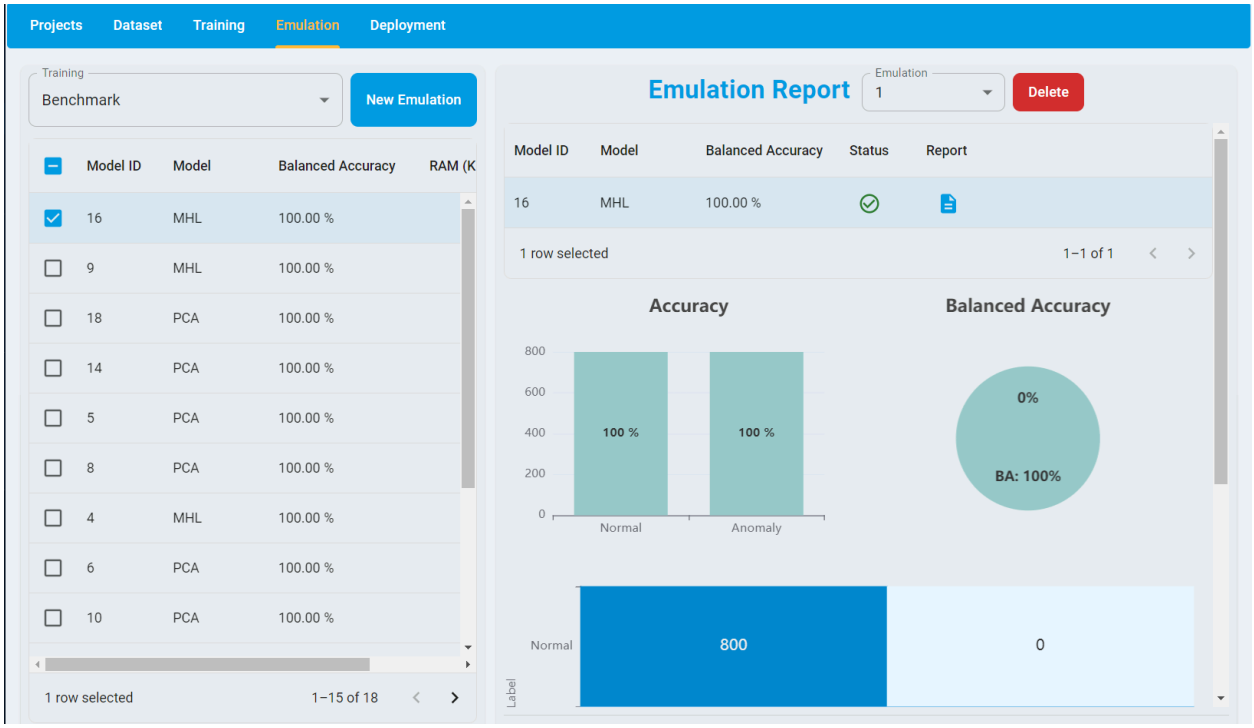

Figure 6 – *Emulation results showing signal similarity and inference response.*

4.5 Deployment to the Microcontroller

1. In the **Deployment** tab, click **Generate** to export the compiled library (`libtss.a`).
2. Extract the files to your project directory (e.g., `C:\Github\nxp_embedded_machine_learning\source\lib`).

3. Confirm that `libtss.a` is correctly placed in the folder.

4. Rebuild the project in **MCUXpresso IDE**.

5. Flash the updated firmware to the i.MX RT1052 board.

In the Tera Term terminal, select **Option 1** to start real-time inference with the embedded model. The system will capture live acceleration data, compute its similarity to the reference behavior, and display the percentage match via the serial port.

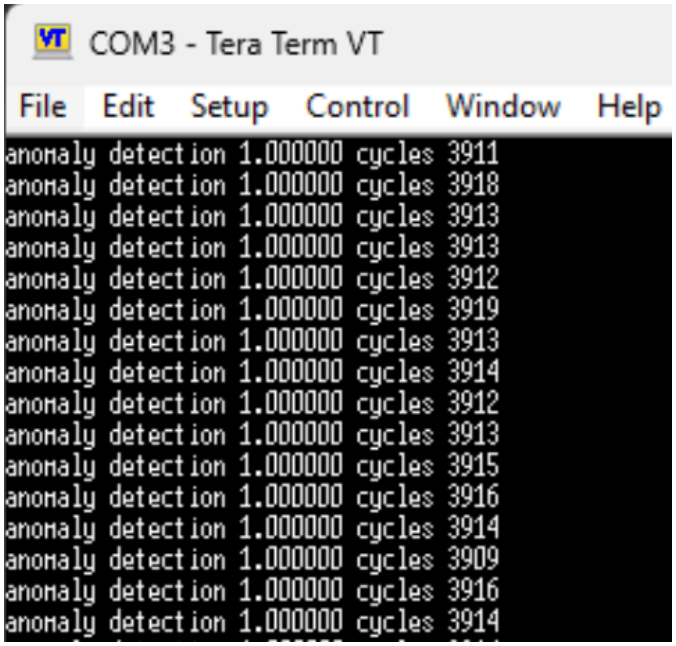

Figure 8 – *Real-time inference output via serial terminal.*