NXP Semiconductors
User's Guide

Document identifier: AN12851
Rev. 0, 14 May 2020

eIQ Anomaly Detection Custom Case Step-by-Step

Contents

Chapter 1
Document overview

This user's guide provides information to replicate setup for AN12766; it includes steps for training the model and converting it to TensorFlow Lite and finally to header file which is exported to application project for i.MX RT1050 board.

This guide also describes minimum package requirements for running python scripts.

Chapter 2
Prerequisites

There are several packages that need to be installed, as per requirement by the python scripts. Python must be installed in system. Other package required by python are mentioned below. Python package installation command may vary depending upon python version installed in system.

1. Update the python installer tool which is called pip. All the command are run on window command prompt.

```
python -m pip install -U pip
```

2. Install the TensorFlow libraries and support for python.

```
python -m pip install TensorFlow
```

3. 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 tensorflow-datasets
python -m pip install xlwt python-tk pandas
python -m pip install numpy scipy matplotlib ipython jupyter     sympy nose
python -m pip install opencv-python
python -m pip install PILLOW
python -m pip install pyserial
```

4. If you are on Windows OS, add the following directories to your executable PATH in windows environment if they are not already:

```
<python install directory>/scripts
```

5. Install vim v8.1 or above for conversion of .tflite file into .h file. If on Windows, install Vim 8.1 from vim.org

   There is a binary convertor programmed named xxd.exe located inside that package that will be needed.

6. If you are using Windows OS, add the following directories to your executable PATH in windows environment, if not already installed.

```
<vim_install_directory>
```

7. Verify that the PATH is set correctly by typing “tflite_convert” and “xxd -v” in a command prompt. You should not get any errors about an unrecognized command.

Chapter 3
Steps for training the model and converting it to TensorFlow Lite format

3.1 Data collection

1. Setup the EVKB i.MX RT1050 on fan to capture its vibrations on the position desired for the trained model. Make sure that the accelerometer (FXOS800CQ) is in place at U32 as shown in Figure 1:

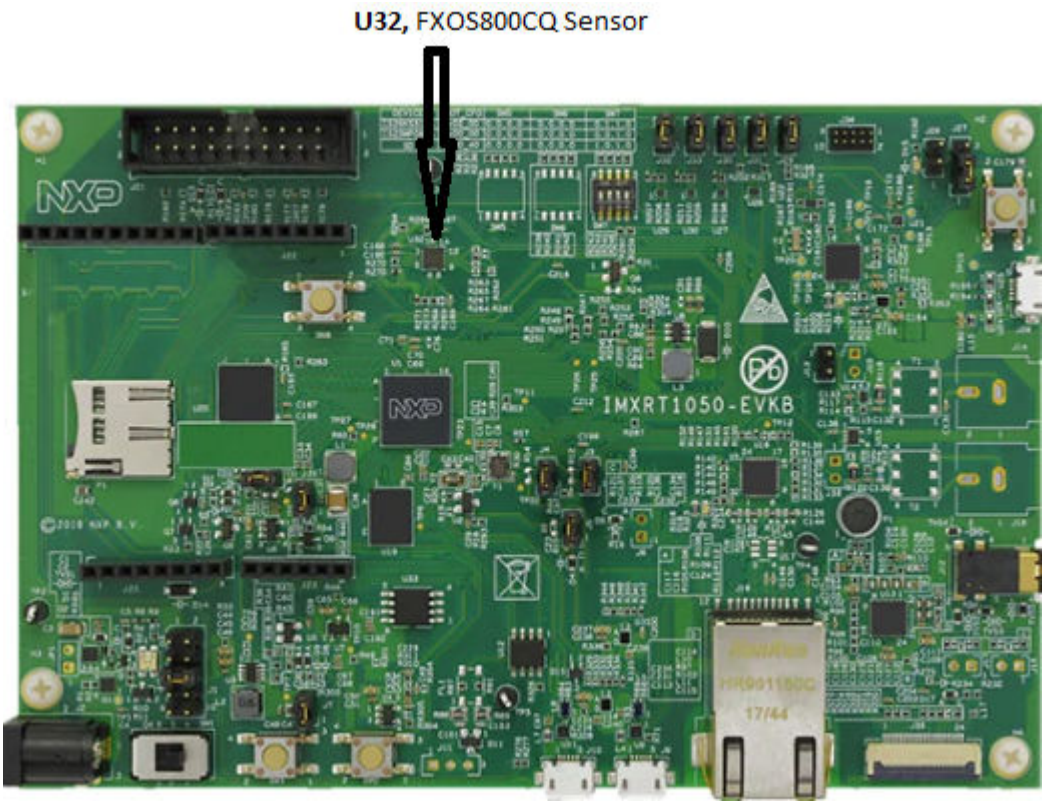


Figure 1. FXOS8700 accelerometer location

2. Flash the application ”evkbimxrt1050_tensorflow_lite_anomaly_detection” with macros disabled **#define RUN_INFERENCE** in timer.h file for data collection and need to enable for running inference in application. Make sure that your serial console is configured with parameter as shown in Figure - 2.

   For Port connection, check your device manager and see which COM port has been assigned. You will start getting X, Y, and Z axis data on serial console.

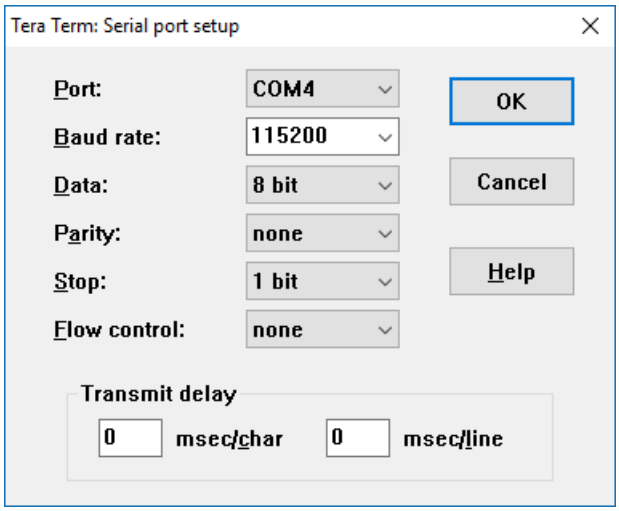


Figure 2. Tera Term serial console configuration

3. Close the console and run the script “Data_Collection.py” from MLPC folder. The script can be modified for the amount of time you want to capture the data as follows:

   a. Hours_ = 1

```
########################################################################
# Timming Configurations
########################################################################
Hours_ = 1
```

Figure 3. Timing configuration

   b. After completion of the configured time period, the script will end and you would see a workbook of the name “Data_Collection_<time>.xls”, in the folder scripts/DataBase.

NOTE

While Data_Collection.py script is running, you need to tap or shake i.MX RT board time to time, so that bad data (anomalous data) also get collected along with healthy data

4. Once Data collection workbook is available in folder scripts/Database, run the “kmeans_development_training.py” from MLPC folder. The complete execution of kmeans_development_training.py script will consume decent time depending upon system configuration.

3.2 Training the model

After you performed the training (after executing the kmeans_development_training.py), you should have 9 trained values, 3 for each X, Y, and Z axis as shown in Figure 4.

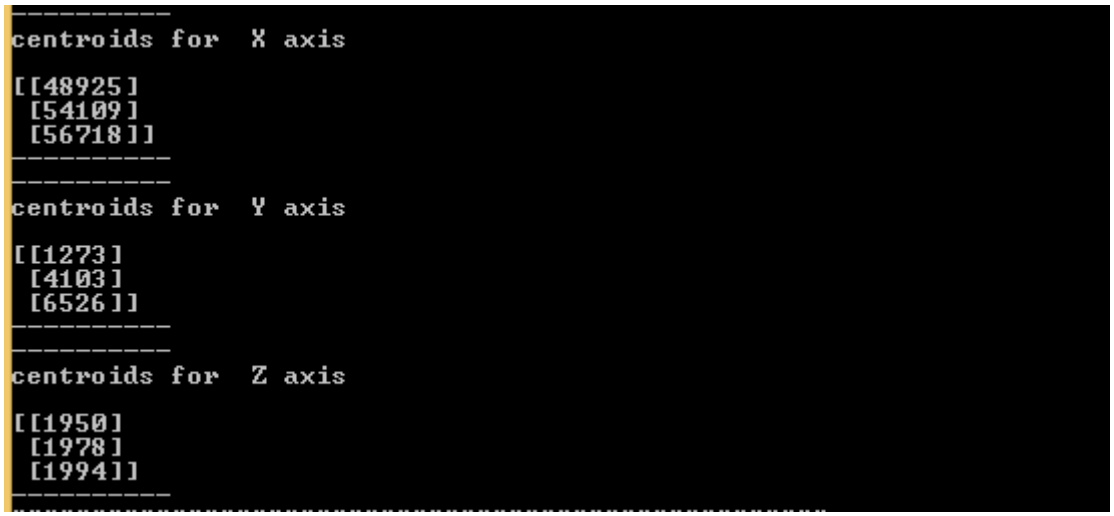


Figure 4. Centroids value for X, Y, and Z axis

3.3 Export the trained model

1. Feed above 9 values (as shown in Figure - 4) to the script “Export_trained_data_model.py” as follows:

```
#x-axis
input_datax = np.array([48925,54109,56718])
#y-axis
input_datay = np.array([1273,4103,6526])
#z- axis
input_dataz = np.array([1950,1978,1994])
```

2. Run the script “Export_trained_data_model.py” from MLPC folder. You will find the “.pb” file in Saved_Model folder as exported trained model which needs to be converted to .tflite. The exported model can be converted to .tflite using the script “conversion.py”.

3.4 Conversion to TensorFlow Lite

1. Run the script “conversion.py” from MLPC folder.
2. You will get a file with .tflite extension in TFlite_Model folder which is the converted model to tensorflowlite format.
3. Open the window command prompt. Now navigate to “..\ML_Release\Scripts\TFlite_Model” folder and execute the following command to convert the .tflite file to .h format.

```
xxd -i ./converted_model_new.tflite > ./converted_model.h
```

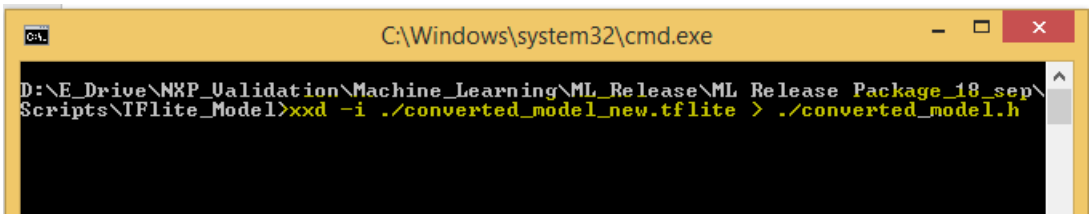


Figure 5. Instruction for model conversion to converted_model.h

4. After converted_model.h file is generated, change “unsigned char” to “const char” in converted_model.h file, as shown in the following figure:

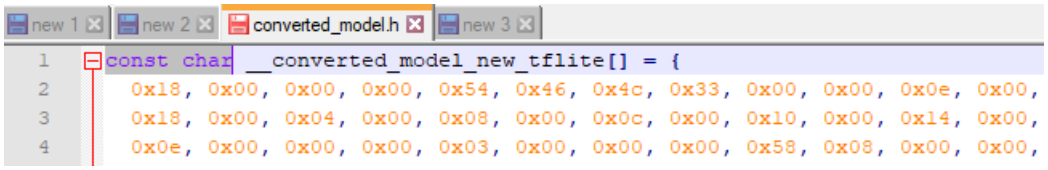


Figure 6. Edited converted model file

3.5 Adding the converted model header file to application project

1. Now add converted_model.h header file in the TensorFlow Lite application project. Your file must look like as shown in the following figure.

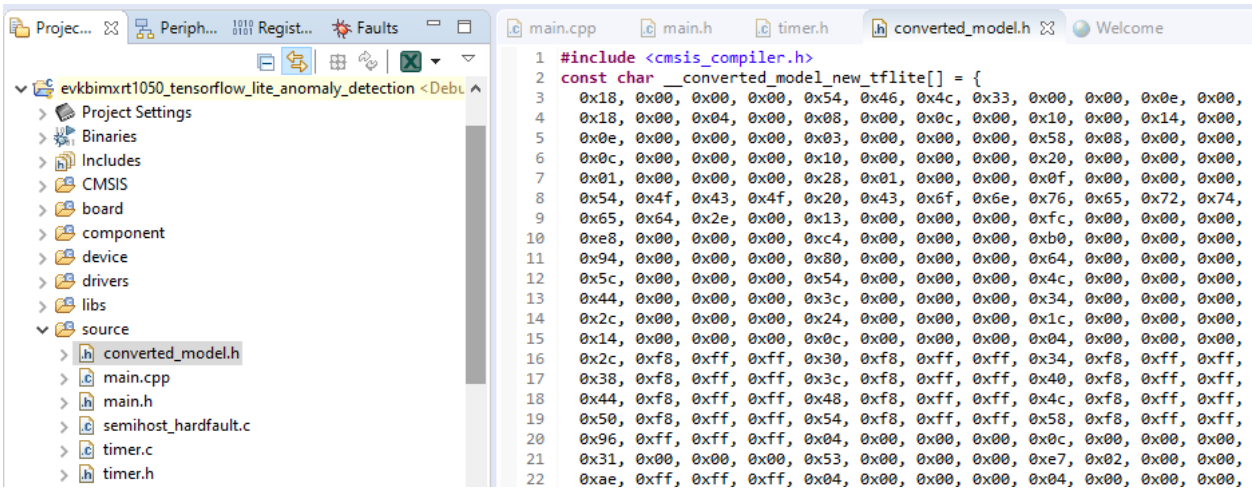


Figure 7. Anomaly detection model file in application project

2. Enable the **#define RUN_INFERENCE** in timer.h file, and replace the model buffer name in main.cpp file as shown below.

```
tflite::FlatBufferModel::BuildFromBuffer(model_data,model_data_len);
```

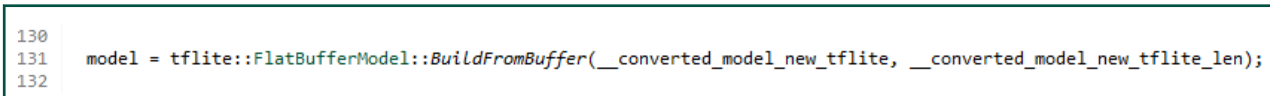


Figure 8. Replace model data buffer name in main.cpp

3. After flashing the binary on i.MX RT1050 board, you will view the output on serial console as shown in the following figure. You can also view output on GUI by running Display.py script.

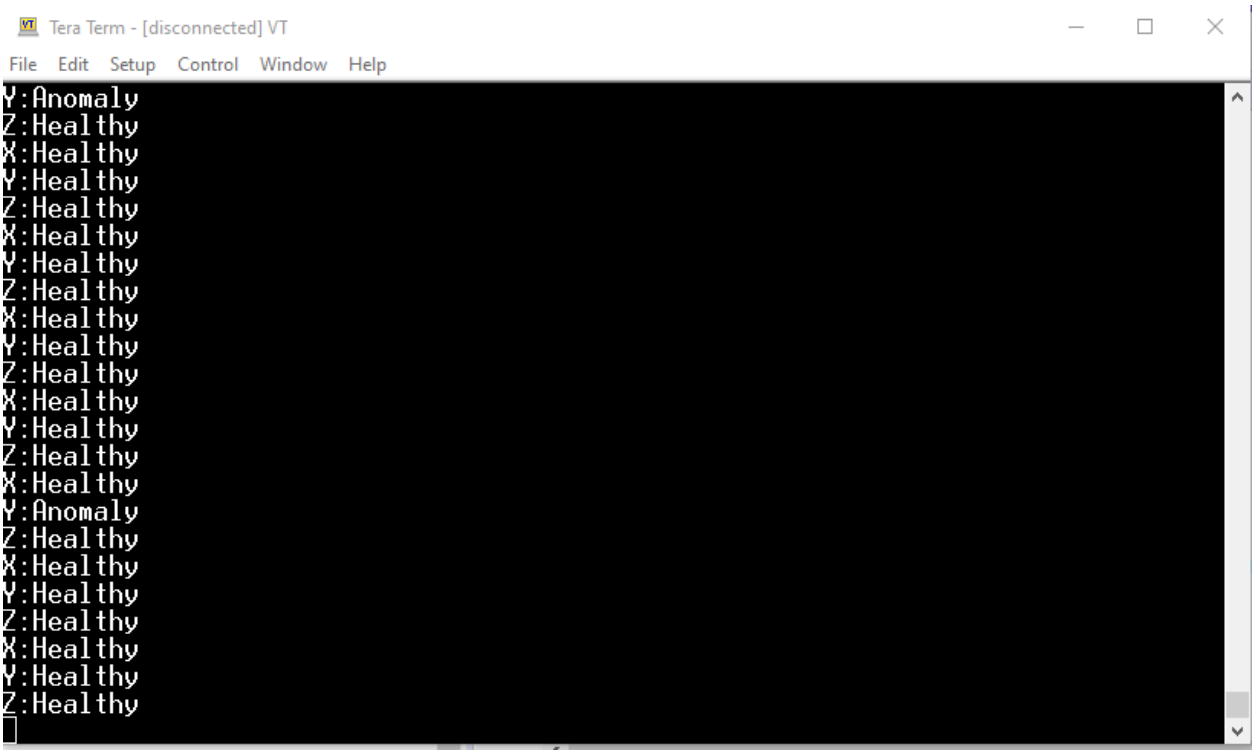
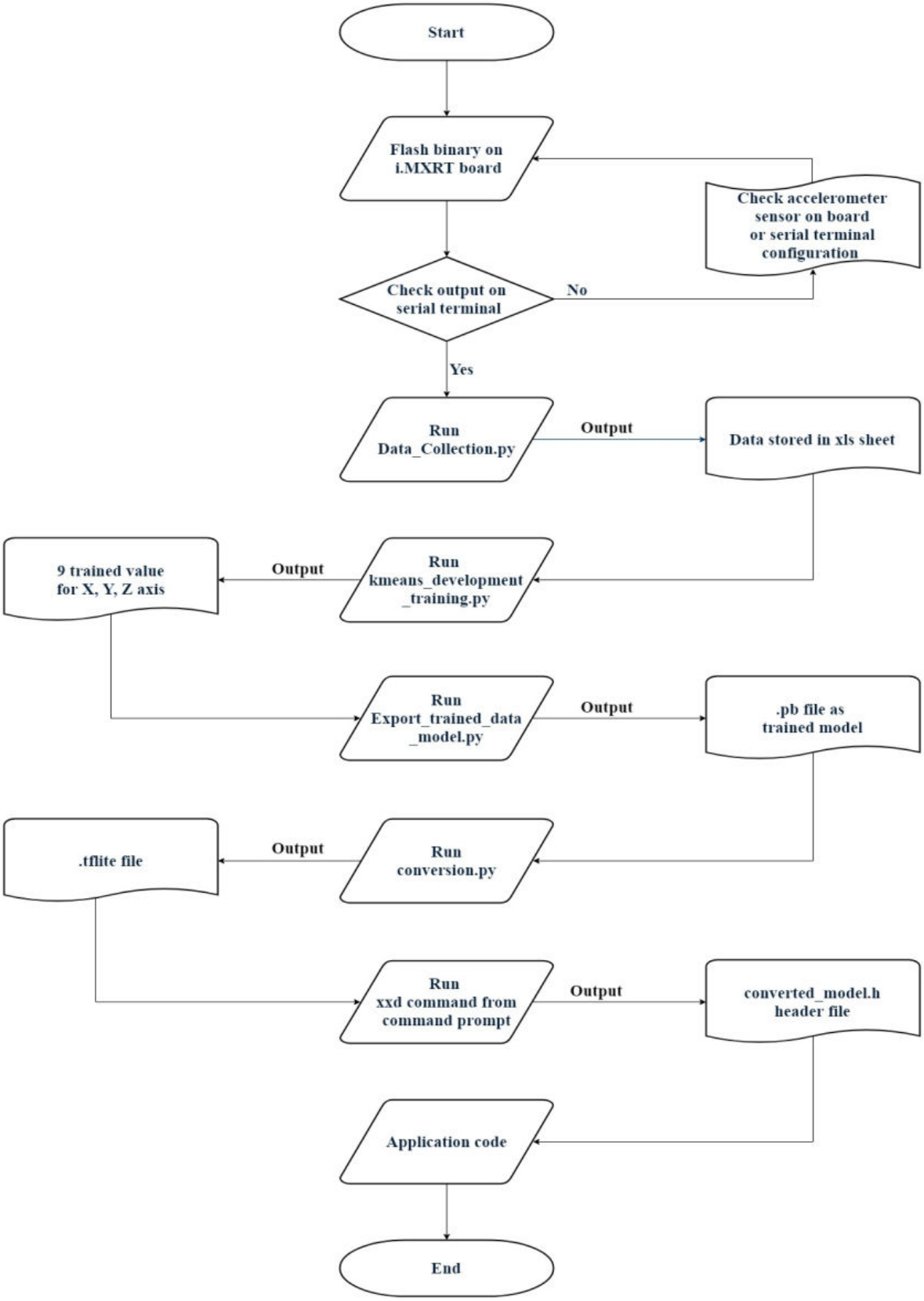


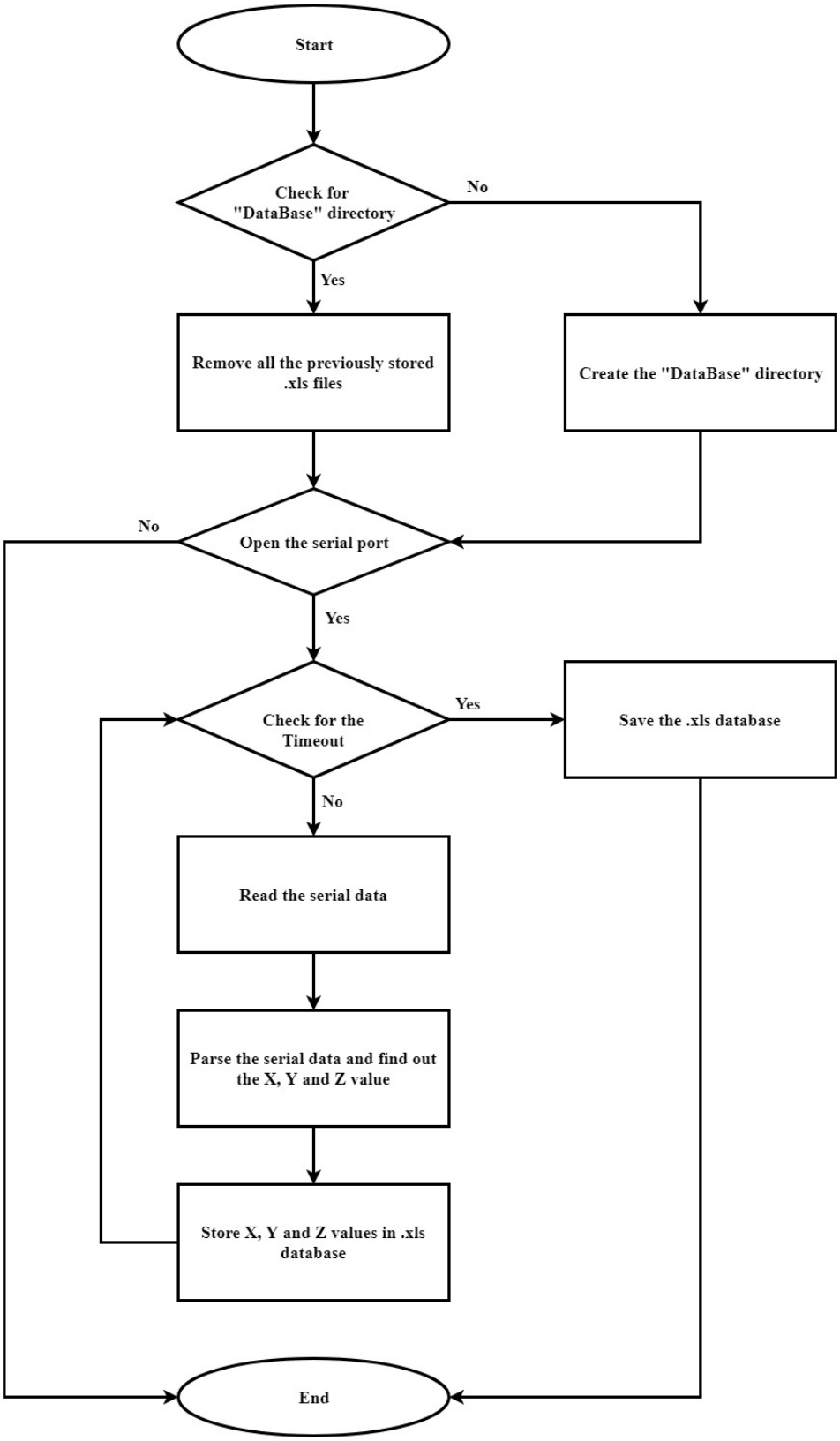
Figure 9. Application output on serial console

Chapter 4
Script description flowcharts

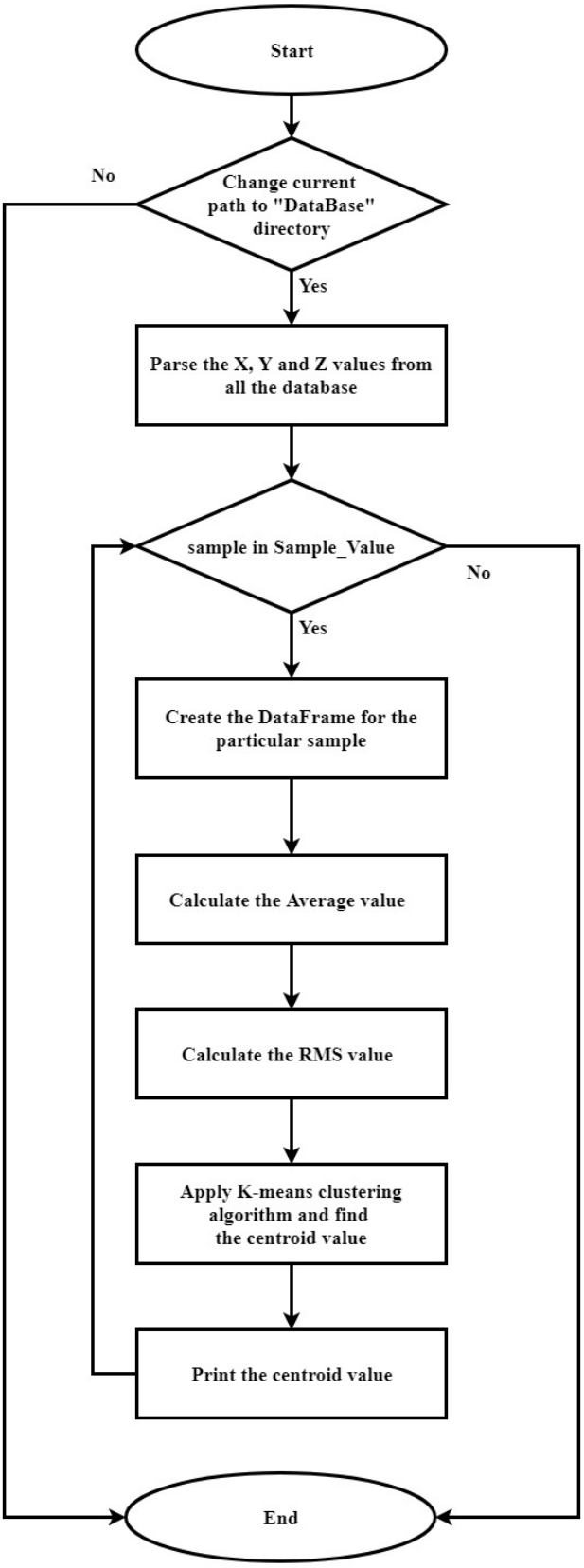
4.1 Flowchart for level transition diagram for model training and its deployment on edge.



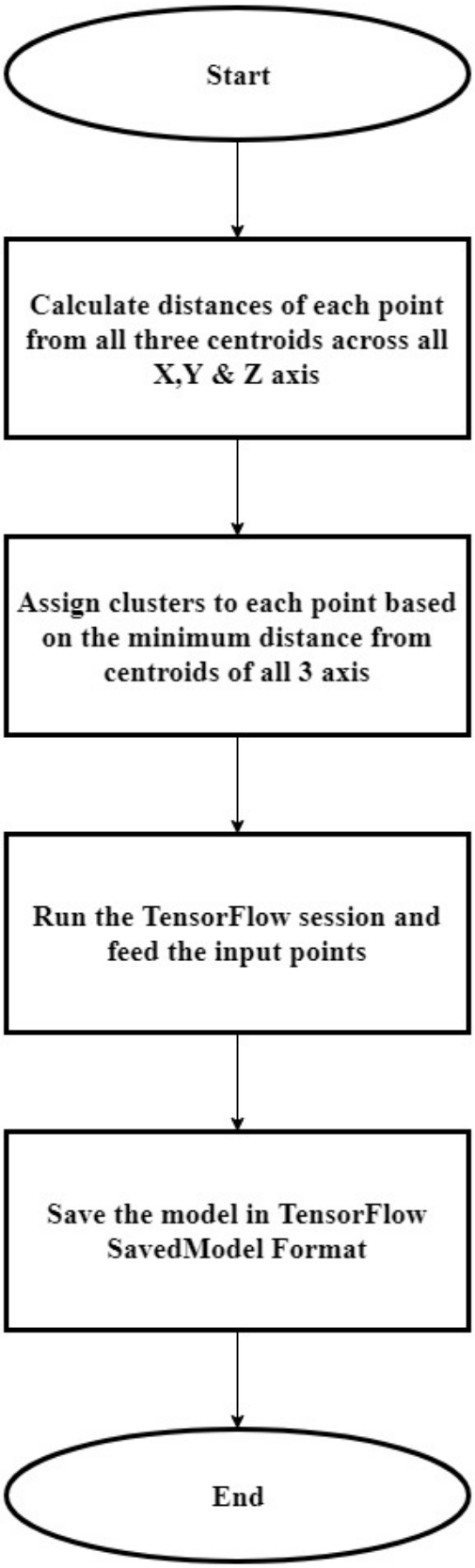
4.2 Flowchart for execution of Data_Collection.py



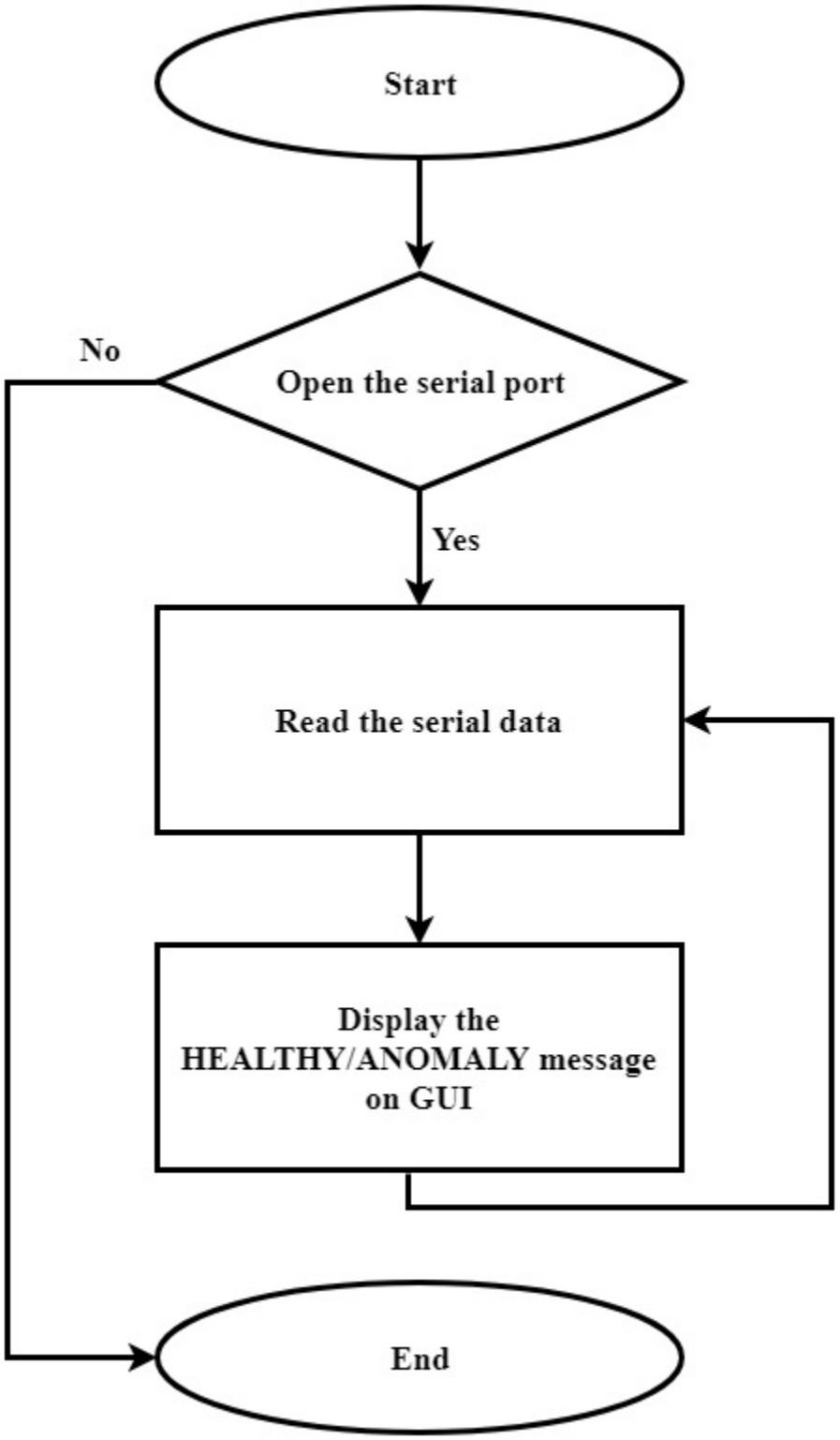
4.3 Flowchart for execution of Kmeans_Development_Training.py



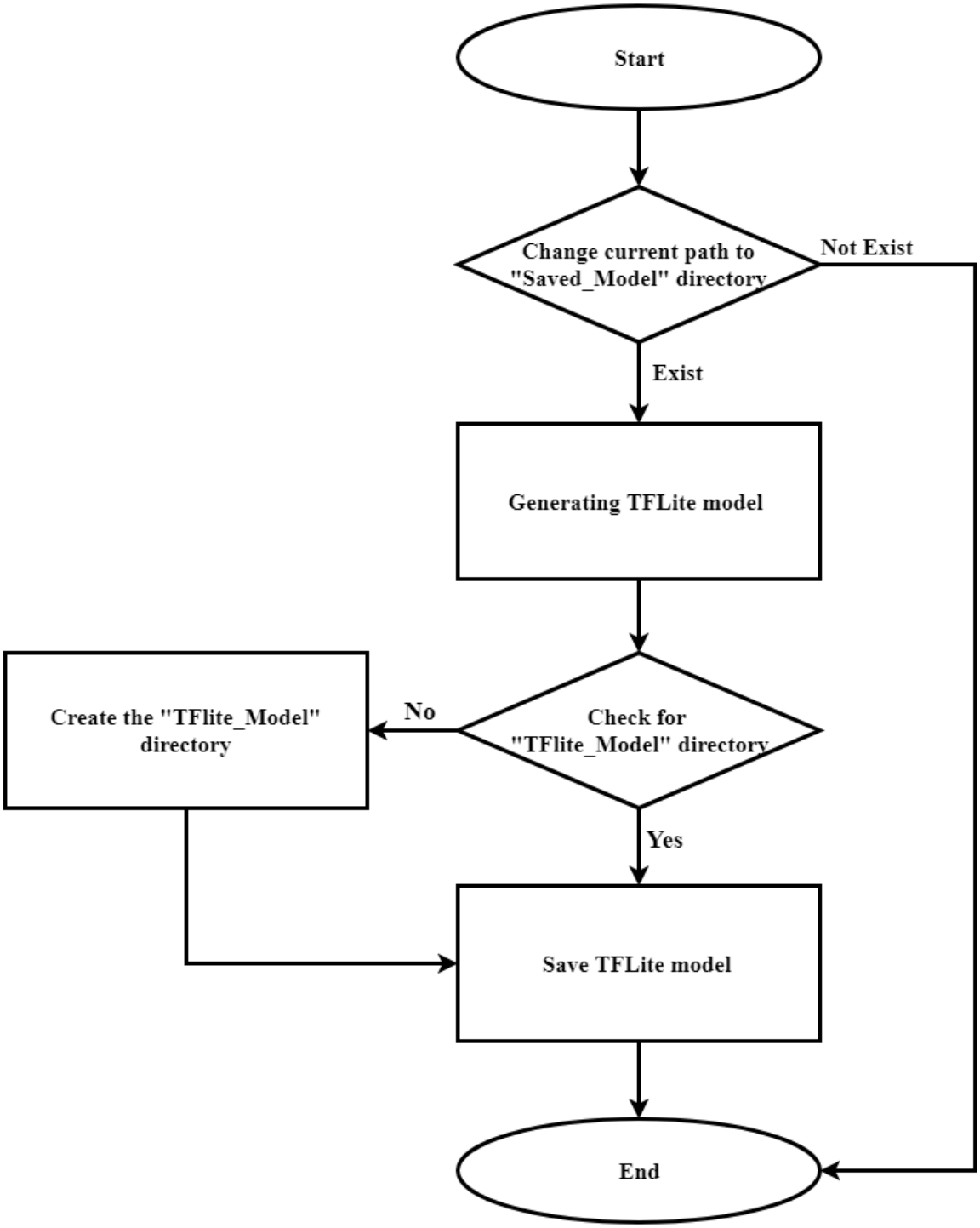
4.4 Flowchart for execution of Export_Trained_Data_Model.py



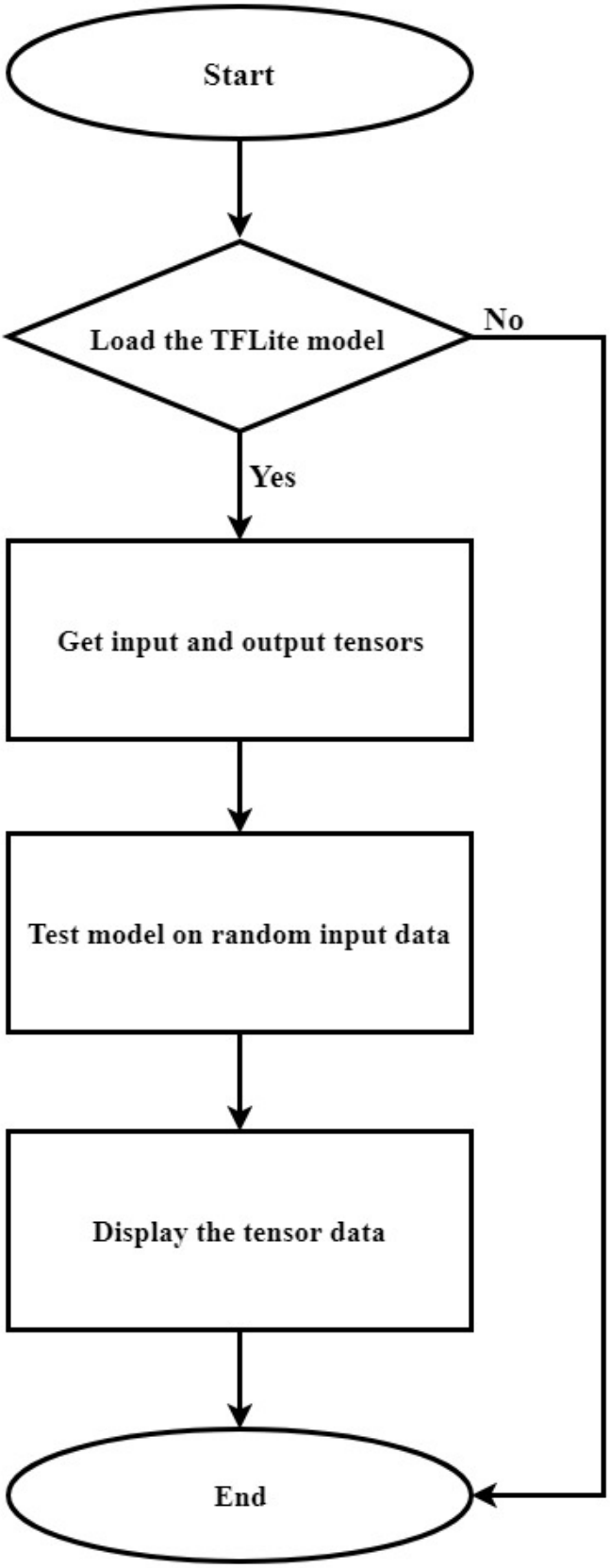
4.5 Flowchart for execution of Display.py



4.6 Flowchart for execution of Conversion.py



4.7 Flowchart for execution of Check_TFlite.py



Chapter 5
Revision history

Rev. Number	Date	Substantative Changes
0	12 May 2020	Initial release

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 14 May 2020

Document identifier: AN12851

arm