Anomaly Detection Toolbox

Rev. 0.6 — 28 April 2017

User manual

Document information

Info	Content
Keywords	ISSDK, Sensors, Sensor Data Analytics, Anomaly Detection
Abstract	Provides full details on the capabilities, structure and use of the anomaly detection toolbox.

Revision History

Rev	Date	Description
<0.1>	2016-09-09	Initial concept level version of the document. Published prior to development of the features described herein, for the purposes of obtaining user feedback. Please send comments to mike.stanley@nxp.com.
<0.2>	2016-09-14	Added screen cartoons and command interface. Added “NXP Internal Notes” section to provide context and product limitations. Removed magnetometers from consideration. Added information w/r to embedded architecture.
<0.3>	2016-09-15	Updated features tab.
<0.4>	2017-01-16	Updated figures for main tab, features tab and GMM tab. Added binary packet definitions for the above.
<0.5>	2017-02-27	Updated serial packet protocol and broke it out into a separate chapter. Updated figures, TOC, etc.
<0.6>	2017-04-04	Added details of controls on each screen, updated all screen shots, added Ensemble models
<0.7>	2017-04-28	Added explanation for GMM and SVM parameters for more details.

Contact information

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

The Table of Contents and Index are at the end of this document.

1. Overview

1.1 NXP Internal Notes

This is a predevelopment version of the user guide for the anomaly detection toolbox. As of this writing, NONE of the components described herein should be assumed to exist. This version of the document is intended to act as a requirements specification to drive development of the toolbox.

This version of the toolkit does not address the problem of concept drift. This will be addressed in a second pass of the toolkit.

The toolkit has a number of assumptions built into it. These generally limit the scope and make the implementation more tractable from both developer and use perspectives. These limitations are:

Primary components of the toolkit are embedded application and associated Windows-based GUI. Embedding machine learning (ML) right into the embedded application allows us to add value to NXP devices.

The user is restricted to one development board – Trying to generalize a higher level ML model within the embedded app is considered unfeasible for the universe of applications. This restriction allows major simplifications in both embedded and GUI implementations.

We reject the use of continuous magnetic calibration (as done in the sensor fusion products) because it can introduce artificial discontinuities when the model is updated. We therefore also reject the use of magnetic sensors as feature inputs.

1.2 Introduction

Anomaly detection encompasses a set of machine learning techniques used to flag when a system departs from normal or expected behavior based upon features extracted from sensor data.

In the context of this document, we discuss a software toolbox targeted at NXP Kinetis MCUs with the following features and capabilities:

- Focuses on single-class anomaly detection. There is no need to acquire data corresponding to abnormal states.
- Utilizes minimally supervised machine learning techniques[1]
    - Gaussian Mixture Models (GMMs)
    - Single Class SVM
    - Kernel variants of the above
    - Adaptive variants of the above
- Embedded Components include:
    - Compatible with the NXP IoT Sensing Software Development Kit (ISSDK):
    - Embedded data logging functions

1. [1] There are other techniques which can be used for anomaly detection. Those listed here are simply the subset we are focusing on with this toolkit.

  - raw data
  - feature data
  - In situ machine learning functions for the techniques listed above
- Graphical user interface for Windows
  - Data logging
  - Monitor & guide learning process
  - Monitor fielded models
  - Report generation
  - Matlab enablement includes functions:
  - to read data files generated by logging functions above
  - to replicate each of the anomaly detection techniques above

The toolkit supports models of a specific structure, which is:

assert anomaly if at least m of n simple models asserts; m<=n

where each simple model is a function of at most two features extracted from the raw sensor data, and m and n are user specified at model configuration time.

The maximum value of "n" is a function of the amount of available RAM.

1.3 Overview

Figure 1 shows the major components in the work flow. These are:

1. Matlab, which can consume raw data files created by the data logging function of the toolbox GUI (2). The use of Matlab is optional for this toolkit.
2. Windows-based control and visualization tool.
3. FRDM-K66F Freedom board - the nominal hardware target.
4. Embedded Software Development Environment for compiling and installing configured models to the Freedom board
5. Model configuration file which represents a specific "hypothesis set" to be used for machine learning.

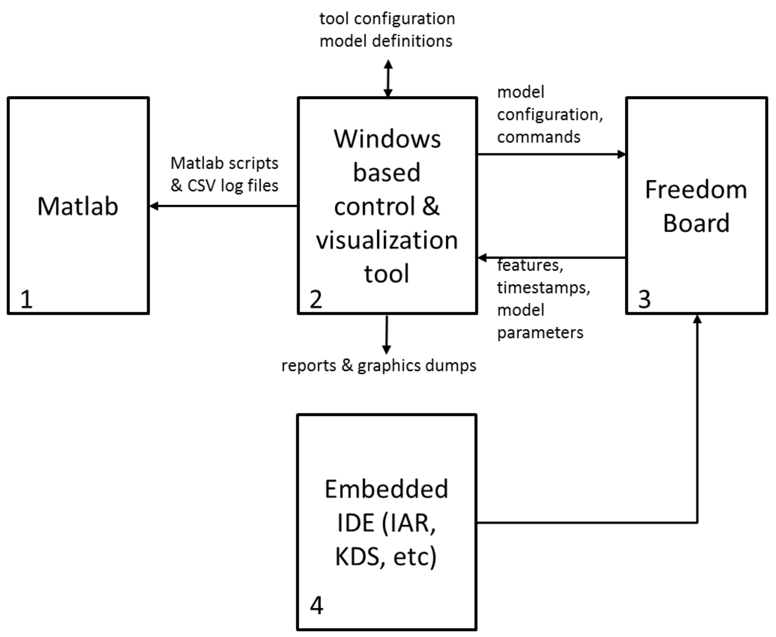

tool configuration
model definitions
Matlab
1
Matlab scripts & CSV log files
Windows based control & visualization tool
2
model configuration, commands
features, timestamps, model parameters
Freedom Board
3
reports & graphics dumps
Embedded IDE (IAR, KDS, etc)
4

Figure 1 Anomaly Detection Toolbox Work Flow

1.4 Use Models

1.4.1 Case 1: In Situ Learning

In this case, it is anticipated that developers use the toolkit GUI on a few test units to determine

- features required
- models required
- failure thresholds

The toolkit then creates a model configuration file, which is used to modify the embedded components. These can then be mass produced such that each manufactured device is capable of learning (using the now specified model type) as a function its unique environment.

The key point here is that all feature extraction and learning algorithms reside on the embedded platform. The GUI is only used to guide selection of learning algorithm parameters.

1.4.2 Case 2: Learn once per product type

The initial phases of this use case are identical to those above. The difference is that once an initial model is learned, that specific learned model will be frozen for all time. Fielded units would all include copies of the same learned model, but would not themselves have the ability to learn.

This use case applies when it has been proven that extracted features behave precisely the same across the population of mass produced devices. In this case, it may be possible to utilize a less expensive MCU, since there is no need for in situ learning.

1.5 Resources

1.5.1 Supporting Documentation

TBD

1.5.2 Training

An excellent graduate level (18 lectures) on machine learning is available from Caltech:

Learning from Data

Professor Yaser Abu-Mostafa

The course covers basic theory behind machine learning and cover several of the techniques used within this toolbox. The course is free and on-demand.

1.6 Hardware & Software Requirements

1.6.1 MCU

Machine learning algorithms are computationally complex, and require significant amounts of RAM. The following minimum requirements are assumed:

- ARM® Cortex®-M4F, 120MHz or greater
- 256KB RAM
- 1MB Flash Memory

1.6.2 Sensors

The minimum complement of sensors required for use of this toolkit is a tri-axial accelerometer.

1.6.3 Communications

The communications physical interface is abstracted such that it should be a straightforward task to create alternate implementations. The initial release is based upon a simple UART-based serial interface.

1.6.4 Software Requirements

The toolbox is written in standard C code. Hardware interfaces are abstracted using the NXP Kinetis Software Development Kit (KSDK).

The library is designed to operate in both bare metal and RTOS environments.

Windows 7 and 10 are supported PC operating systems.

1.6.5 Recommended Platform

The IAR Embedded Workbench is the preferred tool of choice for the embedded portions of the workflow.

The NXP FRDM-K66F Freedom Development Platform (shown below) is the recommended hardware platform for initial development.

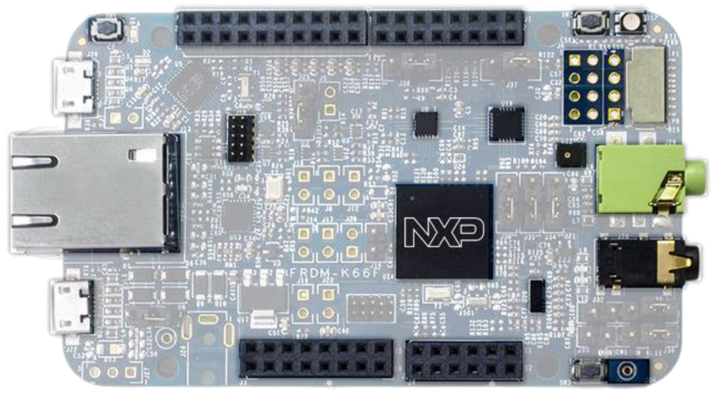


Figure 2 FRDM-K66F Freedom Development Platform

The FRDM-K66F meets all the requirements outlined above, including:

- ARM® Cortex®-M4F, 120MHz or greater
- 256KB RAM
- 2MB Flash Memory
- onboard Ethernet port
- headers for 2.4GHz radio add-on modules
- OpenSDA v2.1 serial interface
- Micro SD card socket
- FXOS8700CQ 6-axis accelerometer plus magnetometer
- FXAS21001 Gyroscope
- Digital MEMS microphone

2. Machine Learning Background

2.1 Introduction

Machine learning is a very broad topic encompassing many different types of algorithms which attempt to learn patterns from data.

This section only provides enough background to help the novice place techniques supported by this toolbox within the boarder context of “machine learning”. Subsequent sections will provide a brief summary of each topic, and then point the reader to more information online.

If you are already knowledgeable on the topic, you can skip this section.

2.2 Supervised vs Unsupervised

ALL machine learning techniques require data from which they learn. In an ideal world, each data element would be labeled with a tag or expected value. Machine learning algorithms then compute a function f() which predicts:

expected value or tag = f (input data)

Linear regression is perhaps the best known example of this. Given known input and measured output values, statistical techniques are used to compute coefficients for the input to output transfer function.

This is the best case scenario. In many cases, we do not have access to known output values for use in training. Consider the case of condition monitoring of factory machinery. We would like to differentiate between “optimal”, “pending failure” and “failure” states. It will be unlikely that the factory operator will be willing to run a statistically significant number of machines to failure just so that you can take data for non-optimal operating states.

Unsupervised learning[2] does not require output values or tags in order to come up with a useful result.

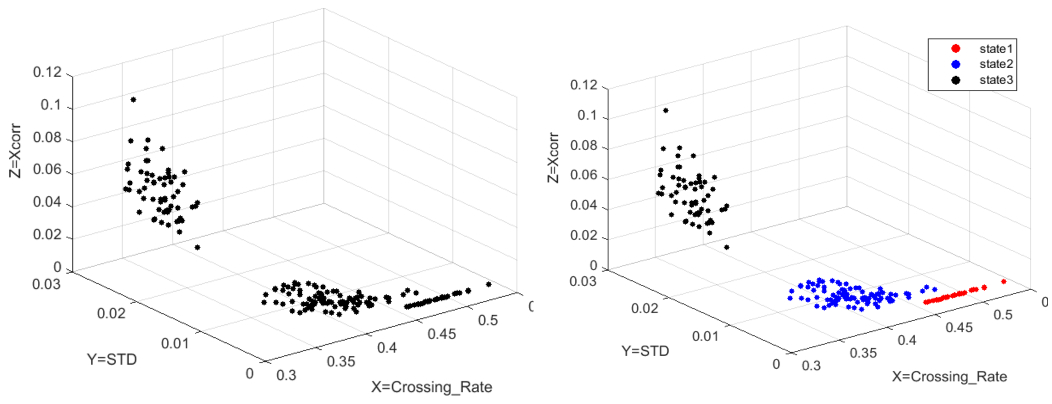


Figure 3 Unsupervised (left) vs Supervised (right) data

The figure above illustrates the difference. Although we can clearly see patterns in the data on the left, we have no idea what those patterns imply. Do we have three different states being displayed, or one or two multimodal states? Tags in the supervised data (right) make it clear that we have data corresponding to three operating states.

2. [2] actually, a better term is: “minimally supervised”

Not discussed here is “reinforcement learning”, which is somewhere between unsupervised and supervised learning. The toolkit does not support this technique.

See https://en.wikipedia.org/wiki/Supervised_learning for additional background.

2.3 Classification vs Regression

If you are computing a continuous function f() which returns a real number as a function of one or more real number inputs, you are performing regression.

If your function f returns 1 of n enumerated types, you are performing classification. Enumerated values might be: “red”, “green” and “blue” or “functional”, “failing” and “failed”. Essentially, they can be any useful tags you like.

Anomaly detection comes under the heading of classification. You want to know if something is operating nominally or not. Some techniques may also give you a probably associated with that determination.

See https://en.wikipedia.org/wiki/Statistical_classification for additional background.

2.4 Gaussian Mixture Models

Gaussian mixture models, or GMMs, model your data as a summation of n-dimensional Gaussian distributions. Each Gaussian is represented by a mean value and covariance matrix.

The figure below shows a two-dimensional slice through the same data we used in Figure 3 above. This run ignored state tags from the input data set. So we can’t tell from the figure whether we have:

one tri-modal distribution; OR

one single-mode and one two-mode distribution; OR

three single mode distributions

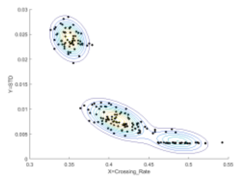


Figure 4 Example Gaussian Mixture Model

In anomaly detection, we treat all input data as coming from a single “known good” distribution. Anything outside of the distribution is assumed “bad”. For the case above, we would choose contour lines associated with some desired variance, and treat everything outside the contours as bad.

Gaussian mixture models have the advantage that they are easy to understand. Anyone who has taken a basics statistics course can easily grasp the essentials.

GMM consists of multiple Gaussian components mixed with a certain ratio. Assume there are M components, A model function f() can be described as

$$f_{GMM}(\text{input data}) = \sum_{m=1}^{M} p_m \, g_m(\text{input data})$$

where $g_m(\text{input data})$ is the m-th Gaussian component with its mean and covariance parameters. The input data is considered as feature vector (e.g., Crossing_Rate and STD in Figure 4.)

Building a GMM is actually equivalent with estimating a set of parameters - # of components (*M*), mixing ratio (p_m), and the mean / covariance of Gaussian component (g_m). This estimation procedure can be performed by an iterative algorithm. One of which is Expectation-Maximization algorithm. Let θ^t be the parameter set at iteration time t. The EM algorithm conducts the following two steps:

i) Estimate θ^{t+1} from θ^t such that $f_{GMM}(\text{input data};\ \theta^{t+1}) \geq f_{GMM}(\text{input data};\ \theta^t)$.

ii) Repeat the step i), until $\theta^{t+1} \approx \theta^t$.

The step i) actually consists of two steps – expectation and maximization. For more details, see https://en.wikipedia.org/wiki/Expectation%E2%80%93maximization_algorithm.

See https://en.wikipedia.org/wiki/Mixture_model for additional background.

Once we build a model function $f_{GMM}(\text{input data})$ with the parameter set θ, then it can represent a probability density function (PDF). Anomaly decision is made with a user defined threshold with the following form:

True (1) if PDF $f_{GMM}(\text{input data}) > \text{threshold}$.

False (-1) otherwise.

$$sgn\ [f_{GMM}(\text{input data}) - \text{threshold}].$$

2.5 Support Vector Machines

Support Vector Machines, or SVMs, are a class of algorithms which attempt to determine optimal hyperplanes to separate multiple classes of data. The algorithms only look at data points nearest neighboring data class clusters to perform the analysis. These data points are referred to as “support vectors”.

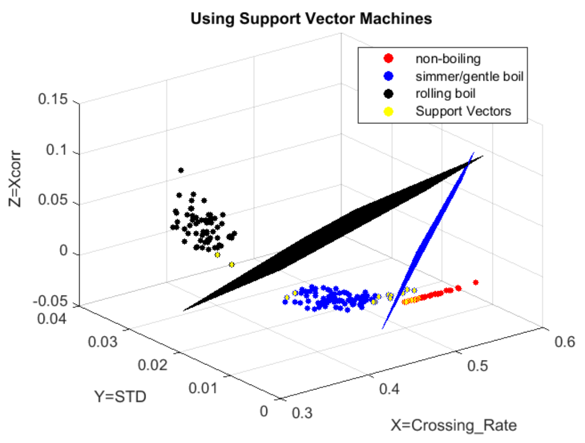


Figure 5   A three state support vector machine

The example shown above is obviously an example of supervised learning. For anomaly detection, there is a variant of the technique referred to as “one-class SVM”.

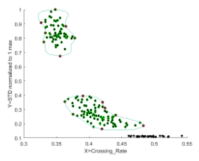


Figure 6 One-Class SVM

0 utilizes the same data set as earlier examples, but in this case, we've artificially collapsed two states and used that as our "normal state" for the analysis. By applying a radial basis kernel and casting the data into a higher dimension, the algorithm is able to do an exceptional job differentiating "normal" from "abnormal".

Note that the decision boundaries include a number of "normal data points". In practice, this is probably undesirable. The algorithm has a number of knobs for us to play with to adjust just how closely the contour tracks the data. We should also be able to expand the contour by the simple expedient of lowering the decision threshold slightly.

SVM can be described by a set of support vectors and their corresponding coefficients. A model for the SVM is described as

$$f_{SVM}(\text{input data}) = \sum_{i=1}^{n} \alpha_i e^{-\frac{\|\text{input data}-sv_i\|^2}{2\sigma^2}}$$

where α_i is the coefficient for support vector, sv_i and n indicates how many support vectors were used. $e^{-\|\cdot\|^2/2\sigma^2}$ denotes the radial basis function (RBF) kernel with the kernel size σ^2. Other types of kernels can be also used.

Building a one-class SVM is achieved by solving quadratic optimization problem. The dual problem is described as

$$\underset{\alpha}{\text{minimize}}\ \frac{1}{2}\sum_{ij} \alpha_i \alpha_j\, \kappa(x_i, x_j) \quad \text{subject to} \quad 0 \le \alpha_j \le \frac{1}{\nu N} \ \text{ and } \ \sum_i \alpha_i = 1$$

where $\kappa(\cdot)$ denotes a kernel function (we primarily used the RBF kernel) and x_i denotes input data.

For $\{x_i\}_{i=1}^{N}$ input data samples for training SVM and a user-defined parameter $0 < \nu \le 1$, solution of the optimization problem results in n patterns (Support Vectors) $\{sv_i,\ \alpha_i\}_{i=1}^{n}$ and a threshold $\rho = \sum_j \alpha_j \kappa(x_j, sv_i)$. There is a guide how to select ν parameter:

i) ν is an upper bound on the fraction of outliers.

ii) ν is a lower bound on the fraction of SVs.

There is asymptotic analysis of the bounds in paper – B. Scholkopf et. Al. “Estimating the Support of a High-Dimensional Distribution,” Journal of Neural Computation, Vol 13, Issue 7, pp. 1443 – 1471, July 2001. See this paper for additional background.

Once we build a model function $f_{SVM}(\text{input data})$ with the SVs $\{sv_i,\ \alpha_i\}_{i=1}^{n}$, then it can represent a probability density function (PDF). Anomaly decision is made with a user defined threshold ρ with the following form:

True (1) if PDF $f_{SVM}(\text{input data}) > \rho$.

False (-1) otherwise.

$$sgn\ [f_{GMM}(\text{input data}) - \rho].$$

See https://en.wikipedia.org/wiki/Support_vector_machine for additional background.

2.6 Kernels

Quoting Wikipedia:

Kernel methods owe their name to the use of kernel functions, which enable them to operate in a high-dimensional, implicit feature space without ever computing the coordinates of the data in that space, but rather by simply computing the inner products between the images of all pairs of data in the feature space. This operation is often computationally cheaper than the explicit computation of the coordinates. This approach is called the "kernel trick".

The kernel trick can be applied to both GMM and SVM approaches, which are the focal point of this project.

See https://en.wikipedia.org/wiki/Kernel_method for additional background.

2.7 Evaluating Performance

The Anomaly Detection Toolbox assumes that all training data represents “nominal behavior”. The goal is to “draw a line” around those nominal data points, and treat anything within that line as nominal”.

Figure 7 shows that the fielded system will classify samples as positive (nominal) and negative (anomalies). Sometimes the classifier will get things wrong, and misclassify results. Thus we have four different ways to parse our results:

TP = True Positive = Nominal behavior that is classified as nominal.

FP = False Positive = Anomalous behavior that is classified as nominal

TN = True Negative = Anomalous behavior that is classified as anomalous

FN = False Negative = Nominal behavior that is classified as abnormal

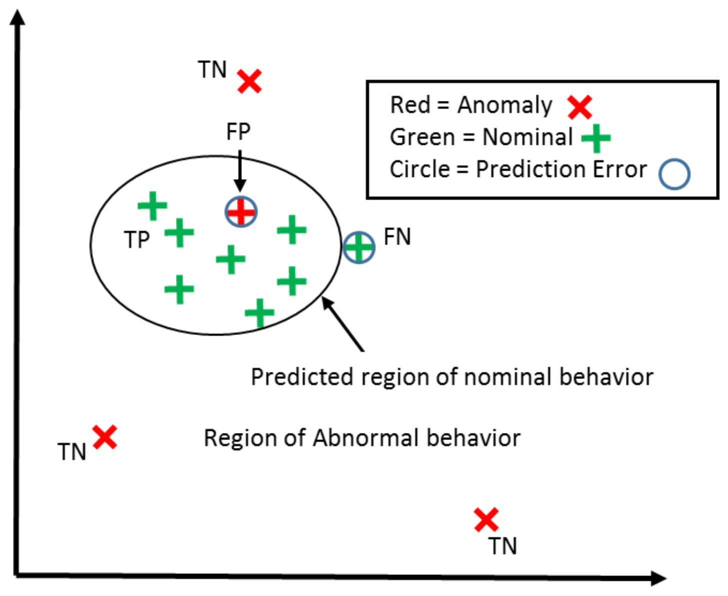

TN
FP
Red = Anomaly ❌
Green = Nominal ➕
Circle = Prediction Error ○
TP
FN
Predicted region of nominal behavior
TN
Region of Abnormal behavior
TN

Figure 7 Anomaly Detection Classification Results

2.7.1 The Confusion Matrix

Machine learning problems are graded via a “decision matrix”:

a.Actual	b.Predicted Positive	c.Predicted Negative
PP = TP + FN = Positive Population	TP = True Positive	FN = False Negative
NP = FP + TN = Negative Population	FP = False Positive	TN = True Negative

Table 1. Confusion Matrix Components

Ideally, all test cases lay on the diagonal. However machine learning is a statistical process, and there will almost always be off-diagonal components. Recurring false negatives result in a “boy who cried wolf” scenario. False positive results are missed opportunities to service equipment before it becomes a critical issue.

Standard Metrics derived from the confusion matrix are:

Accuracy = 1 – Error = (TP + TN) / (PP + NP) = probability of a correct classification

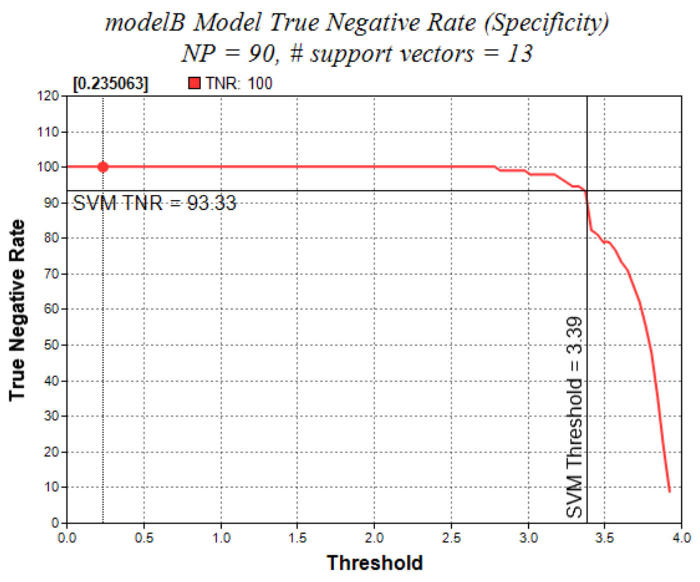
Sensitivity = TP / PP = the ability of the test to detect a positive element from a **positive population**

Specificity = TN / NP = the ability of the test to correctly rule out a condition from a condition free population.

2.7.2 ROC and SENSITIVITY Curves

The Receiver Operating Characteristic (ROC) curve plots Sensitivity on the Y axis versus (1-specificity) on the X axis – effectively true positives versus false positives. It has been used for evaluating learning effectiveness of multiclass machine learning algorithms.

Since we assume that our training population is drawn solely from the positive population, traditional ROC curves are of no use to us. Instead, we will plot TN/NP (specificity) on the Y-axis versus Threshold on the X-axis. The figure below shows an example for a Support Vector Machine. The SVM engine computed a PDF threshold of 3.39, which yielded an effective SENSITIVITY of 93%.



6. Anomaly learning curves as a function of algorithm parameters

2.7.3 Computation Metrics

Future versions of this document will provide guidance on the following as a function of feature set and algorithm types used:

Flash memory requirements

RAM requirements

MIPS requirements

2.8 Ensemble Models

Ensemble models represent another level of hierarchy in terms of a machine model. Each ensemble model is composed of multiple lower level models (GMM or SVM) that effectively “vote” to determine whether a collection of features should be considered nominal or anomalous.

There are numerous variations on this theme. The Anomaly Detection Toolbox supports simplified options discussed in the next two subsections.

2.8.1 Voting

The figure below illustrates a system with three regions of “nominal behavior”. These are shown on the same XY axes for visualization purposes, but may in fact be characterized using different choices of XY features.

These regions might correspond to different modes of operation. Operation characterized as within the regions is normal. Anything outside is considered an anomaly.

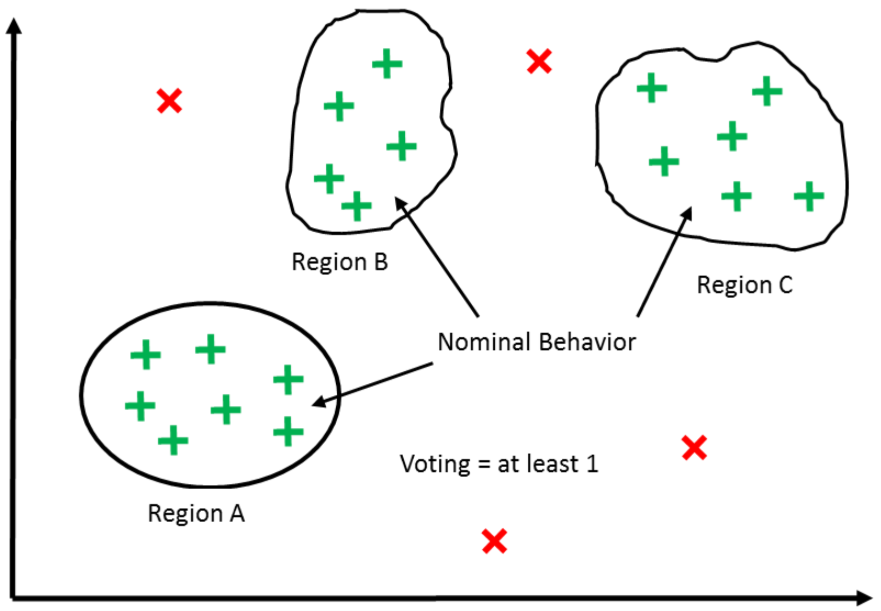


Figure 8 A System with Multiple Regions of “Nominal Behavior”

Assume three lower level models are trained individually on each of the three regions. Then a system anomaly can be defined as the case where none of the lower level models identify as nominal. Put another way, the system is behaving nominally if ANY of the three regions assert.

The figure below uses the same logic to effectively enlarge a single nominal region. One model might be a GMM and the second an SVM.

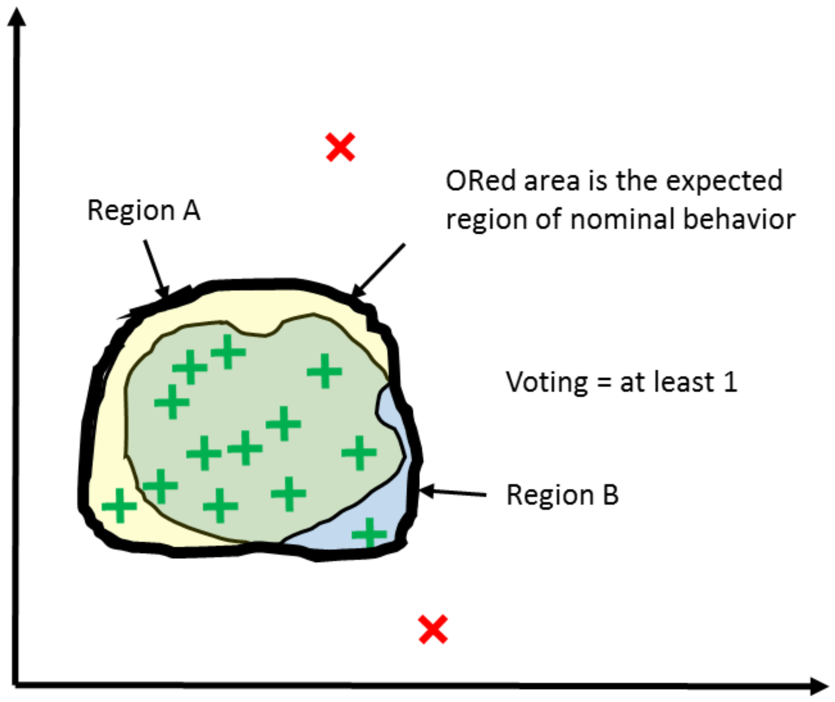


Figure 9 Logical OR of Nominal Regions

If you would light to tighten your model (as opposed to loosening it as in Figure 9), a logical AND function makes more sense. At least 2 lower level models must assert nominal for the system to be judged in nominal condition.

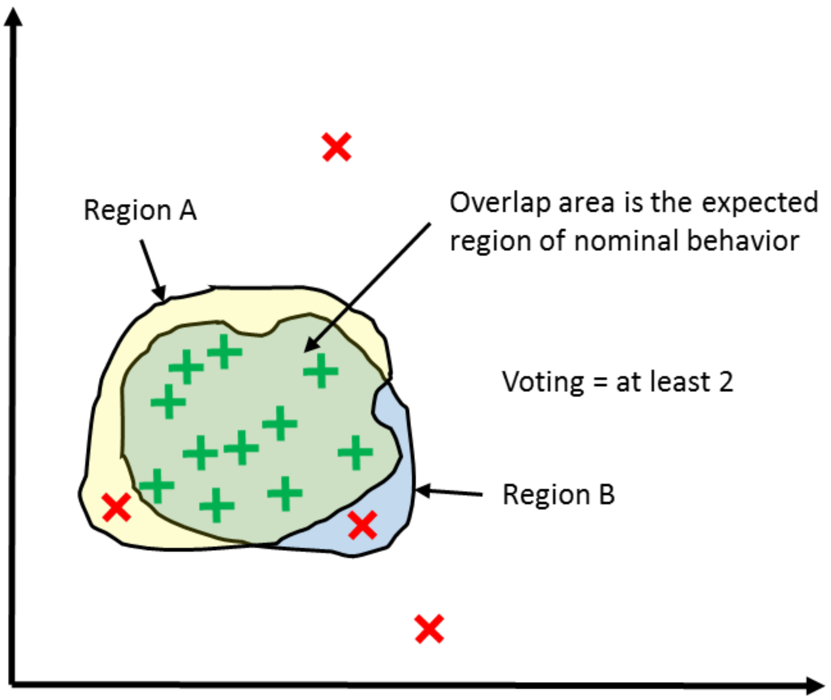


Figure 10 Logical AND of Nominal Regions

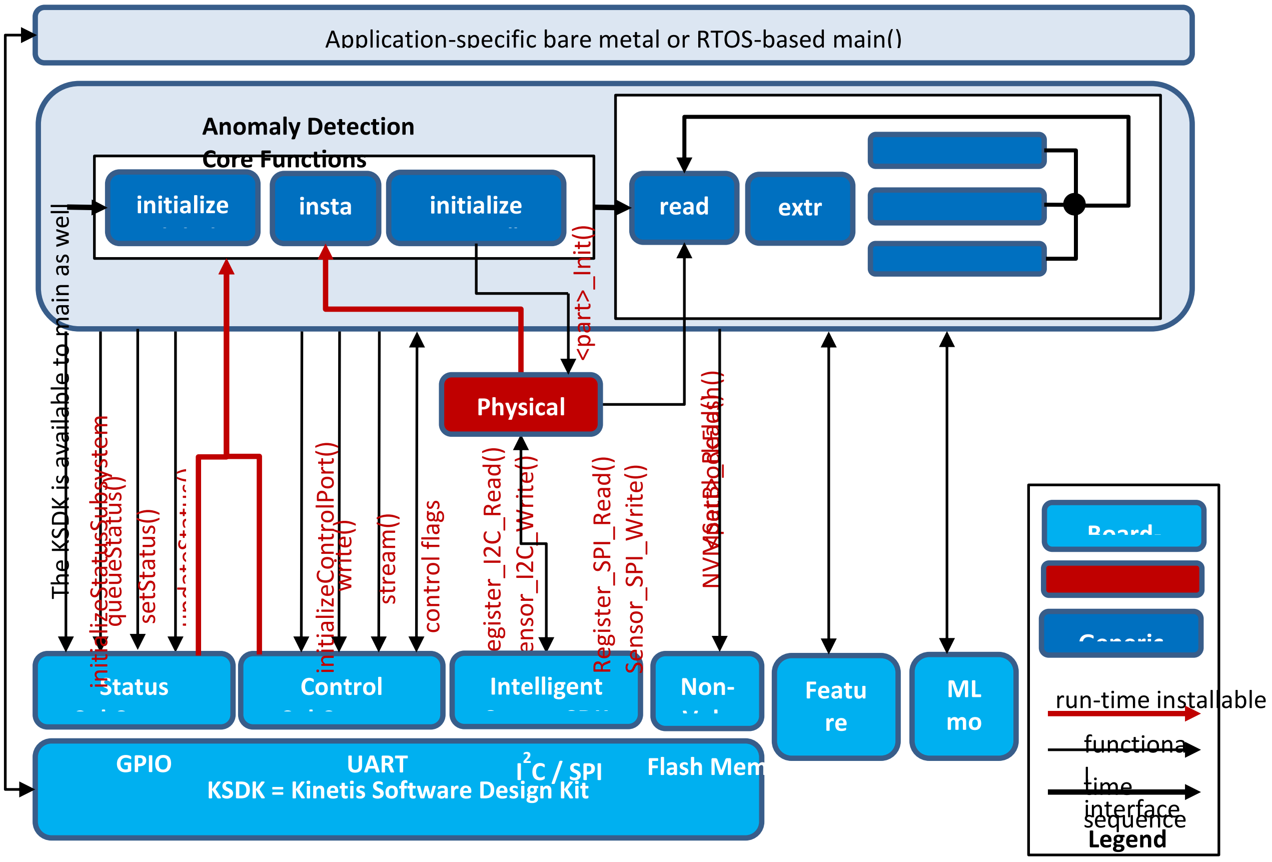
3. Embedded Components

As noted in Section 1.4, all feature extraction, learning and execution of learned model are done in situ on the embedded device. Toolkit functions can be used in either a bare metal or an RTOS environment. FreeRTOS is the RTOS of choice for this toolkit, although the user is free to use any RTOS they desire.

Therefore embedded components include:

- ISSDK
- Status subsystem
- Communications subsystem
- Control sequencer
- Nonvolatile storage functions
- Sensor drivers
- Sample buffers
- Feature buffers
- Statistical feature extraction for:
- FFT and total FFT energy (Phase II)
- standard deviation, variance, skew factor and kurtosis
- mean crossing rate

- …
- Model computation for GMM and One-State SVM
- Execution models for GMM and One-State SVM



-

Figure 11 High Level View of the Embedded Architecture

4. Graphical User Interface

Item (2) in Figure 1 represents the GUI for this project. This tool has multiple views::

- Main screen
- Features screen
- FFT screen (Phase II)
- Screens for GMM, SVM and Ensemble models

Each of these is explored in more depth in the following sub-sections.

Screen displays are mockups and/or rough drafts, and will evolve as requirements evolve.

4.1 Main Screen

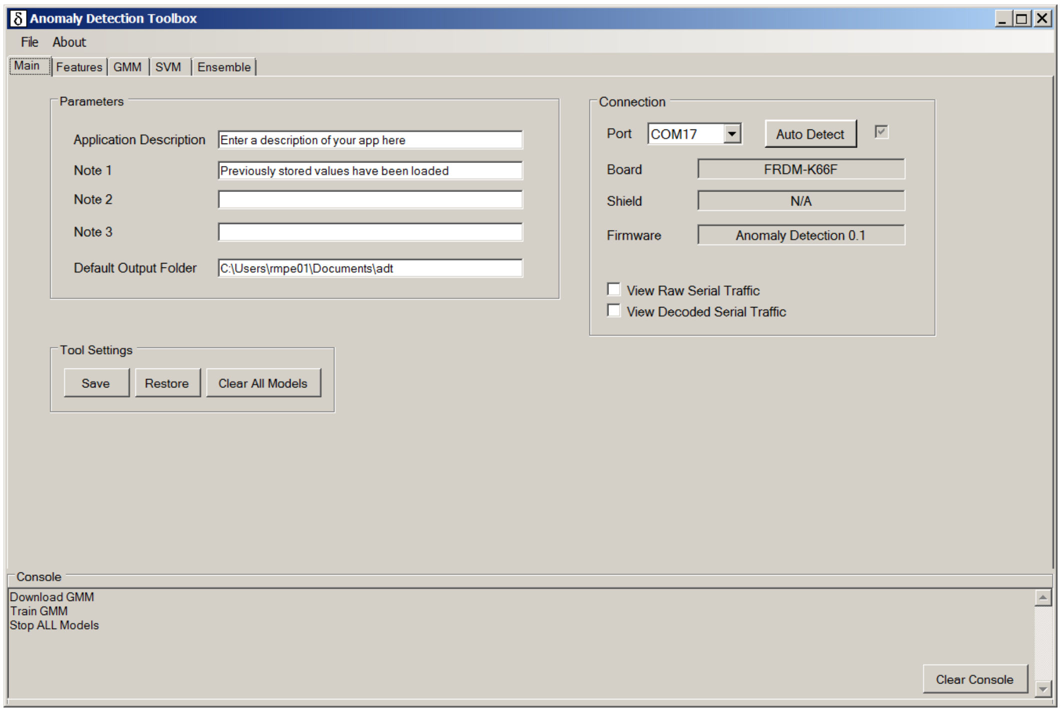
4.1.1 Function

The model screen contains functions for establishing communications to a sensors board, specifying metadata for a various tool runs and confirming hardware and firmware versions.

There are four regions included on the main screen. These are:

- Parameters: Fields for application description, notes and default output folder
- Connection: contains controls for connecting to and validating an embedded board, as well as enabling a display of low level serial communications.
- Tool Settings: Allows you to save/restore the current state of the GUI, as well as clear all ML models from memory (both in the GUI and on the embedded board)
- Console: This region is visible from all screens. It provides a place for the program to communicate detailed status and communications. It can be cleared at any time by pressing the "Clear Console" button to the right.

4.1.2 Screen Dump



Application Description	Enter a description of your app here
Note 1	Previously stored values have been loaded
Note 2	
Note 3	
Default Output Folder	C:\Users\rmpe01\Documents\adt

Port	COM17
Board	FRDM-K66F
Shield	N/A
Firmware	Anomaly Detection 0.1

☐ View Raw Serial Traffic
☐ View Decoded Serial Traffic

```
Download GMM
Train GMM
Stop ALL Models
```

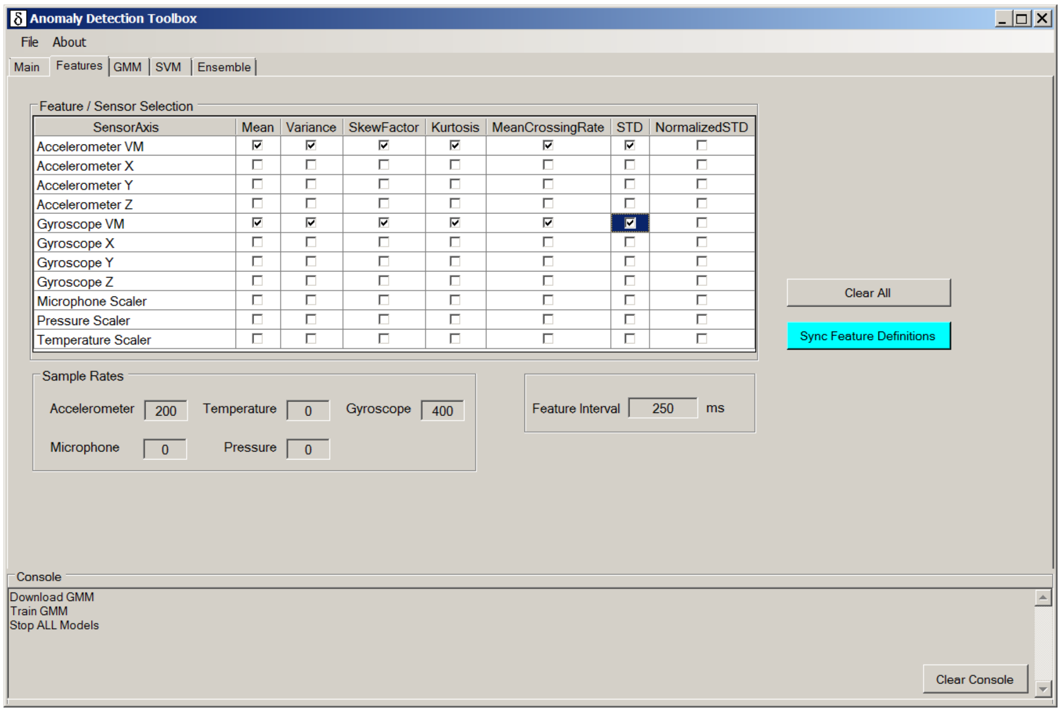
Figure 12 Main Screen

4.2 Features Screen

4.2.1 Function

The features screen lets the user know what sensors are available, and what their sampling rates are. The latter is fixed for a given build of the embedded application. The user is also able to specify their window length in milliseconds. This same period is applied to all features, regardless of their sampling rate. Consider the case of a 1000ms window. The standard deviation for a gyro sampling at 800Hz will be based on those 800 samples, versus 400 samples for an accelerometer being sampled at 400Hz.

4.2.2 Screen Dump



Feature / Sensor Selection

SensorAxis	Mean	Variance	SkewFactor	Kurtosis	MeanCrossingRate	STD	NormalizedSTD
Accelerometer VM	☑	☑	☑	☑	☑	☑	☐
Accelerometer X	☐	☐	☐	☐	☐	☐	☐
Accelerometer Y	☐	☐	☐	☐	☐	☐	☐
Accelerometer Z	☐	☐	☐	☐	☐	☐	☐
Gyroscope VM	☑	☑	☑	☑	☑	☑	☐
Gyroscope X	☐	☐	☐	☐	☐	☐	☐
Gyroscope Y	☐	☐	☐	☐	☐	☐	☐
Gyroscope Z	☐	☐	☐	☐	☐	☐	☐
Microphone Scaler	☐	☐	☐	☐	☐	☐	☐
Pressure Scaler	☐	☐	☐	☐	☐	☐	☐
Temperature Scaler	☐	☐	☐	☐	☐	☐	☐

Sample Rates

Accelerometer 200 Temperature 0 Gyroscope 400

Microphone 0 Pressure 0

Feature Interval 250 ms

Console

```
Download GMM
Train GMM
Stop ALL Models
```

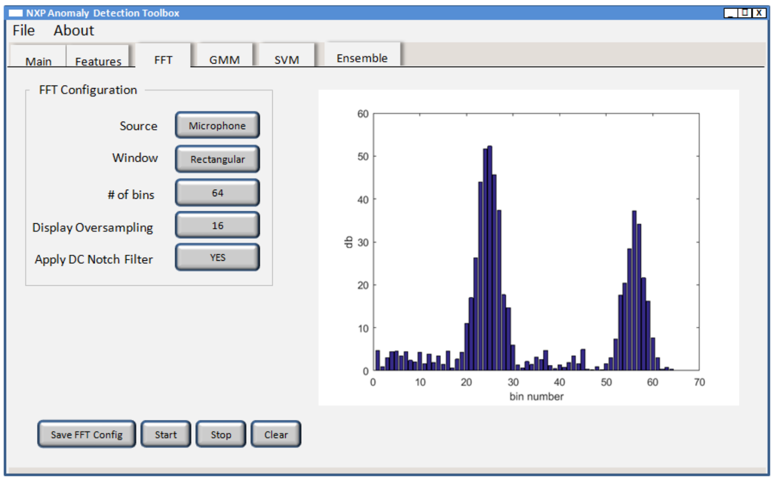
Figure 13 The Features Screen

This screen also lets the user select which features he/she will want to leverage in subsequent toolkit steps.

As shown above, only 1st level features are extracted by the tool. That is, there are no “features of features”. This may change in future versions of the tool.

4.3 The FFT Screen (Phase II)

The FFT screen provides a real-time view of FFT coefficients for the specified sensor input. It also gives the user the ability to specify how many FFT bins are needed and what filtering is required.



Source: Microphone
Window: Rectangular
# of bins: 64
Display Oversampling: 16
Apply DC Notch Filter: YES

Figure 14 The FFT Screen

4.4 GMM Model Configuration Screen

4.4.1 Function

The GMM model screen includes:

- Configuration block, which includes controls for:
  - GMM Model: Select “new” to start defining a new model. Otherwise select from list of previously generated Gaussian Mixture Models.
  - Name: This field is read only unless the model selection is “new”. In which case, use it to specify a name by which you will identify your model in the future.
  - Model Dimension: currently, only 2D models are supported
  - Model Number: This is a read-only field. It contains a unique model number assigned once you click the “Create/Modify” button to save a model configuration.
  - Maximum number of Gaussians: This places a maximum number on the number of Gaussian components that the ML algorithm is allowed to consider.
  - Feature Buffer Size: The maximum number of samples to use for training. The embedded code has a circular buffer that can hold up to 200 of the most recent feature samples. You can choose to use any or all of that to train your model. Using more samples increases model accuracy and generality.
  - Compute interval: The Anomaly Detection Toolbox was designed with a continuous training mode so that you can observe the model evolve as more feature samples arrive. This control sets the time interval between those training events in units of “feature intervals”. The latter is specified on the Features Screen.
  - Threshold X 100: This controls the threshold for the GMM model. You will probably want to use the SENSITIVITY output tab to help you determine where to set the threshold.
  - X and Y Axis Feature: The X and Y Axis Feature pull-downs allow you to select from any of the features previously enabled in the Features screen.

    - Sync Features: You must click this button after making any changes to the list of selected features to update the choices offered in the X and Y Axis Feature pulldowns. This button performs the identical function to the "Sync Features" button on other tool screens.
- Control block, which includes the following action buttons:
    - Create/Modify: Use this to create a new or modify an existing model in the GUI memory.   If form fields have changed since creation, the GUI may give you a hint that the model needs to be updated by changing the button background to.
    - Download: Once a model has been created or modified in GUI memory, it must be downloaded to the development board before it can be trained. The background to this button is changed to cyan to indicate that a download is required..
    - Train: Starts training the model. Feature samples returned to the GUI as marked as "Training Data" to include that they could not be classified, as the model is not yet available.
    - Run: Stops training and starts running the new trained model. Feature samples returned to the GUI are marked as "nominal" or "anomaly".
    - Stop: Stops all model training and run operations. This applies to all models, not just the one currently displayed.
    - Delete Model: Delete the currently displayed model.
    - Clear History: Clears the GUI's history of feature samples for the current model. Since these are used to set min/max values in a number of the plots, clicking this control will both delete points from the plots as well as rescale the plots.
- Model output tabs, which include:
    - 2D Plots
    - 3D Surface Plots
    - Sensitivity Plot
    - Model coefficients

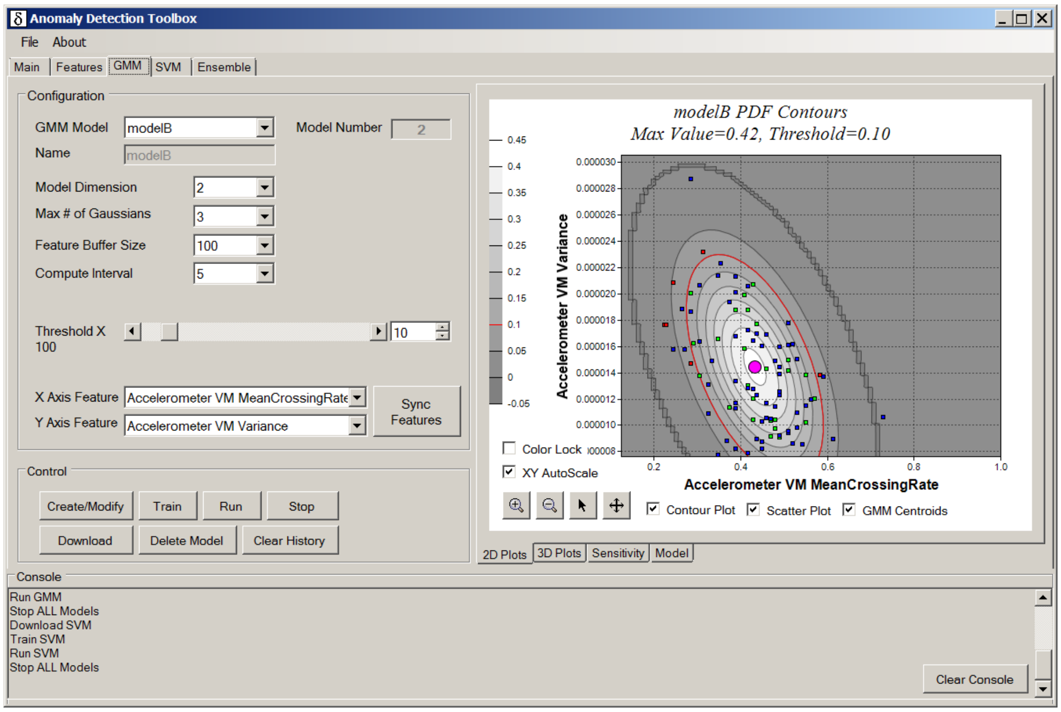
4.4.2 Screen Dumps and Output Controls

The red line shown in the 2D and 3D plots in the next two figures represents the decision threshold for the model. Everything inside those contours is considered nominal. Everything outside those contours is considered an anomaly.

Both plot types have additional controls.   The 2D plots include controls for:

- Zoom In: Changes mouse operation to allow you to zoom the display about a desired region.
- Zoom Out: Changes mouse operation to allow you to zoom out.
- Pan: Changes mouse operation to allow you to click and drag a portion of the plot so as to reposition it's range.
- Expand to full Zoom Out: Zooms X and Y axes out to the full range defined by XY data points in GUI memory. You can reset this range by pressing the "Clear History" button in the Control panel.

- Color Lock: Clicking this control allows you to freeze the current color scale. This is useful when you are in train mode and do not wish to be distracted by continuous updates to the color scale.
- XY AutoScale: Unchecking this control turns off the autoscale feature for X and Y axes, similar to what the color lock control does for the color scale.
- Contour Plot: Set to checked to display model contours. Unchecked to turn them off.
- Scatter Plot: Set to checked to display XY features. Unchecked to turn them off.
- GMM Centroids: When checked, violet dots are drawn at the XY location of each Gaussian component.



GMM Model: modelB
Model Number: 2
Name: modelB
Model Dimension: 2
Max # of Gaussians: 3
Feature Buffer Size: 100
Compute Interval: 5
Threshold X 100: 10
X Axis Feature: Accelerometer VM MeanCrossingRate
Y Axis Feature: Accelerometer VM Variance

modelB PDF Contours
Max Value=0.42, Threshold=0.10

☐ Color Lock
☑ XY AutoScale
☑ Contour Plot ☑ Scatter Plot ☑ GMM Centroids

```
Run GMM
Stop ALL Models
Download SVM
Train SVM
Run SVM
Stop ALL Models
```

Figure 15 GMM Screen – 2D Plots Tab

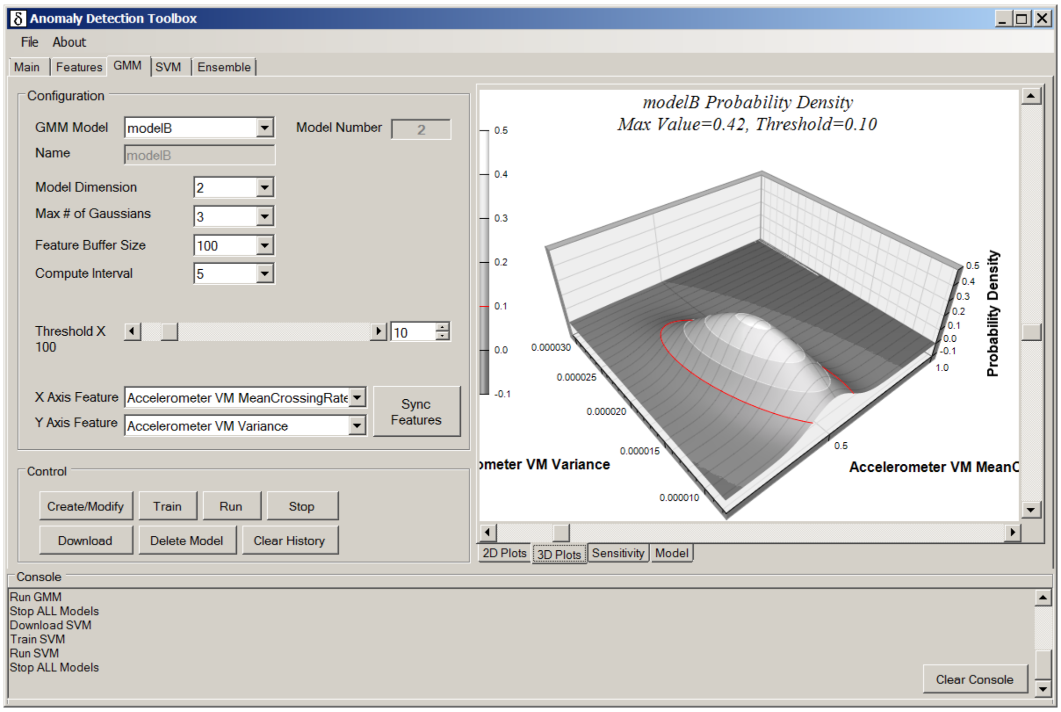
Scatter plot XY points plotted on the 2D plots subscribe to the following color code:

Blue	Used for XY points used for training
Green	Used for points labelled as nominal by a trained model
Red	Used for points labelled as anomalies by a trained model

3D plots include scrollbars along the bottom and right sides of the plot that can be used to adjust the viewing angle of the 3D plot.

Both plot types display the maximum PDF value computed for the XY range, along with the model threshold and model name as part of the plot title.

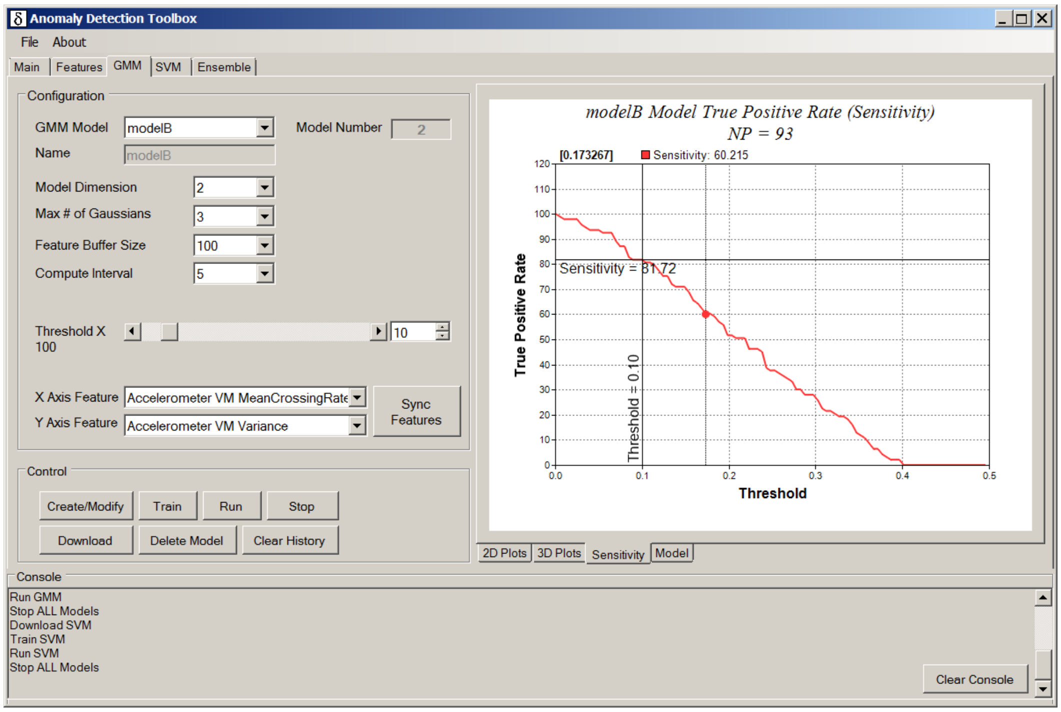
All plots in the Anomaly Detection Toolbox can be dumped to .png graphic files. Simply bring the plot to the foreground and then use the File->Save->Image. The default location for plot files is within the "Default Output Folder" specified on the Main screen of the tool.



GMM Model: modelB
Model Number: 2
Name: modelB
Model Dimension: 2
Max # of Gaussians: 3
Feature Buffer Size: 100
Compute Interval: 5
Threshold X 100: 10
X Axis Feature: Accelerometer VM MeanCrossingRate
Y Axis Feature: Accelerometer VM Variance

```
Run GMM
Stop ALL Models
Download SVM
Train SVM
Run SVM
Stop ALL Models
```

Figure 16 GMM Screen – 3D Plots Tab



GMM Model: modelB
Model Number: 2
Name: modelB
Model Dimension: 2
Max # of Gaussians: 3
Feature Buffer Size: 100
Compute Interval: 5
Threshold X 100: 10
X Axis Feature: Accelerometer VM MeanCrossingRate
Y Axis Feature: Accelerometer VM Variance

```
Run GMM
Stop ALL Models
Download SVM
Train SVM
Run SVM
Stop ALL Models
```

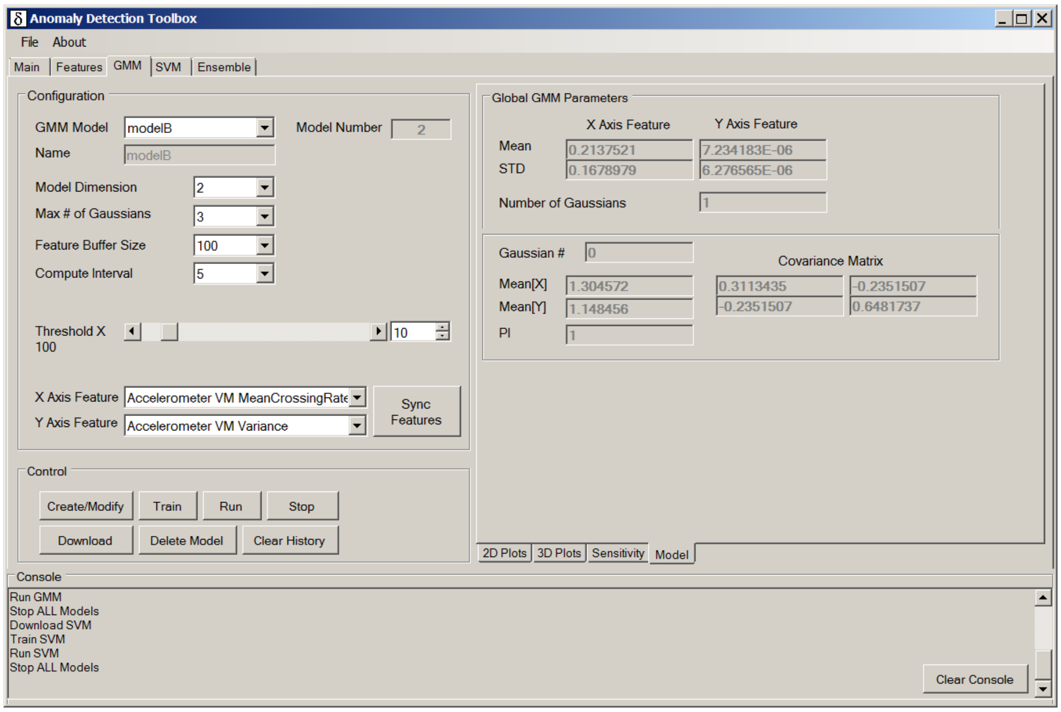
Figure 17 GMM Tab – Sensitivity Plot

The Sensitivity tab gives access to an interactive plot that allows you to determine how good a job your model does identifying nominal feature samples. A vertical guide is drawn at the threshold you specified when you created your GMM model (SVM thresholds are computed by the model itself). The corresponding horizontal guide gives you the percentage of nominal XY features that were classified as nominal.

A cursor guide allows you to read specific XY values from the curve.

Finally, the model tab lets you monitor model coefficients.

All four displays (2D Plots, 3D Plots, SENSITIVITY and Model) update in real time during training. The 3D Plots and Model displays are frozen once the training phase is done. The other two displays continue to update as new XY feature points are received.



Configuration

GMM Model: modelB Model Number: 2
Name: modelB
Model Dimension: 2
Max # of Gaussians: 3
Feature Buffer Size: 100
Compute Interval: 5
Threshold X 100: 10
X Axis Feature: Accelerometer VM MeanCrossingRate
Y Axis Feature: Accelerometer VM Variance

Global GMM Parameters

	X Axis Feature	Y Axis Feature
Mean	0.2137521	7.234183E-06
STD	0.1678979	6.276565E-06

Number of Gaussians: 1

Gaussian #: 0

		Covariance Matrix	
Mean[X]	1.304572	0.3113435	-0.2351507
Mean[Y]	1.148456	-0.2351507	0.6481737
PI	1		

Console

```
Run GMM
Stop ALL Models
Download SVM
Train SVM
Run SVM
Stop ALL Models
```

Figure 18 GMM Screen – Generated Model

4.5 The SVM Screen

4.5.1 Function

The SVM model screen is very similar to the GMM model screen. It includes:

- Configuration block, which includes controls for:
  - SVM Model: Select "new" to start defining a new model. Otherwise select from list of previously generated Gaussian Mixture Models.
  - Name: This field is read only unless the model selection is "new". In which case, use it to specify a name by which you will identify your model in the future.
  - Model Dimension: currently, only 2D models are supported
  - Model Number: This is a read-only field. It contains a unique model number assigned once you click the "Create/Modify" button to save a model configuration.
  - Lower Bound #SVs: This is a read only field. It contains a lower bound for the number of support vectors to be computed for your model.
  - Feature Buffer Size: The maximum number of samples to use for training. The embedded code has a circular buffer that can hold up to 200 of the most recent feature samples. You can choose to use any or all of that to train your model. Using more samples increases model accuracy and generality.
  - Compute interval: The Anomaly Detection Toolbox was designed with a continuous training mode so that you can observe the model evolve as more

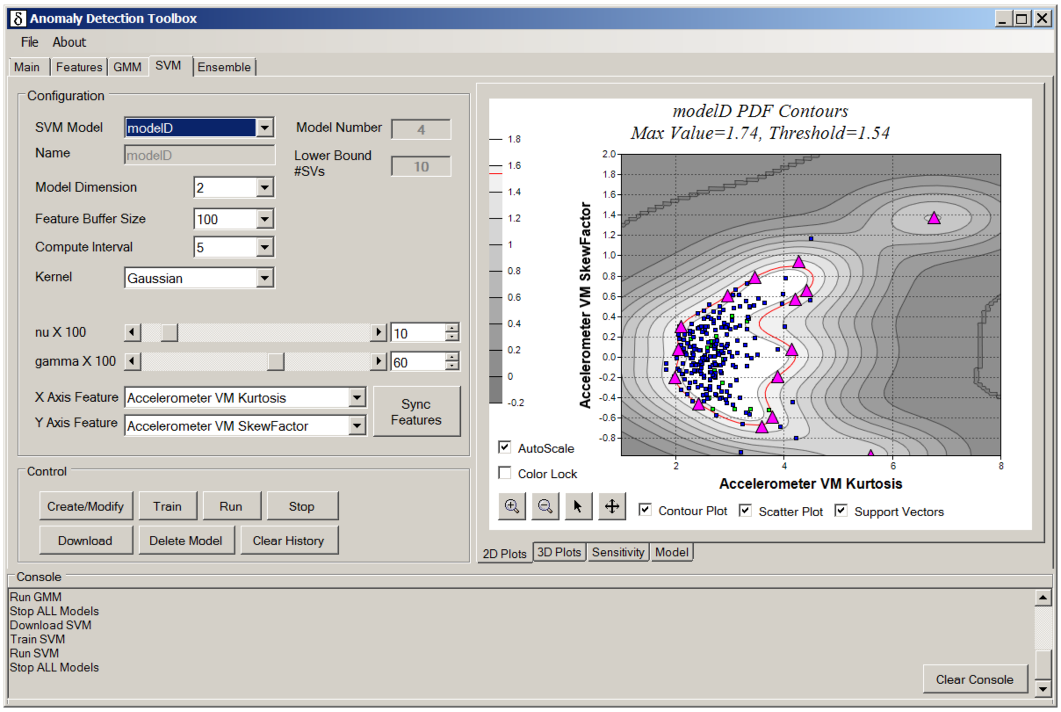
feature samples arrive. This control sets the time interval between those training events in units of "feature intervals". The latter is specified on the Features Screen.

- Kernel: Specifies the Kernel type used for the Support Vector Machine. "Gaussian" is the only currently supported Kernel type.
- nu X 100: Nu in one class SVMs represents the upper bounds of the fraction of outliers and lower bound on the fraction of support vectors.
- Gamma X 100: The square root of gamma is inversely proportional to alpha in the libsvm one-class SVM formulation. Use a larger gamma to "tighten" the model.
- Sync Features: You must click this button after making any changes to the list of selected features to update the choices offered in the X and Y Axis Feature pulldowns. This button performs the identical function to the "Sync Features" button on other tool screens.
- X and Y Axis Feature: The X and Y Axis Feature pull-downs allow you to select from any of the features previously enabled in the Features screen.

- Control block (identical to the GMM control block), which includes the following action buttons:
  - Create/Modify: Use this to create a new or modify an existing model in the GUI memory. If form fields have changed since creation, the GUI may give you a hint that the model needs to be updated by changing the button background to.
  - Download: Once a model has been created or modified in GUI memory, it must be downloaded to the development board before it can be trained. The background to this button is changed to cyan to indicate that a download is required.
  - Train: Starts training the model. Feature samples returned to the GUI as marked as "Training Data" to include that they could not be classified, as the model is not yet available.
  - Run: Stops training and starts running the new trained model. Feature samples returned to the GUI are marked as "nominal" or "anomaly".
  - Stop: Stops all model training and run operations. This applies to all models, not just the one currently displayed.
  - Delete Model: Delete the currently displayed model.
  - Clear History: Clears the GUI's history of feature samples for the current model. Since these are used to set min/max values in a number of the plots, clicking this control will both delete points from the plots as well as rescale the plots.
- Model output tabs are the same as those used in the GMM display. They include:
  - 2D Plots
  - 3D Surface Plots
  - Sensitivity Plot
  - Model coefficients

4.5.2 Screen Dumps and Output Controls

The 2D plot tab is almost identical to that used for GMMs. The one difference is that instead of a control to display Gaussian centroids, we have one to display SVM support vectors.



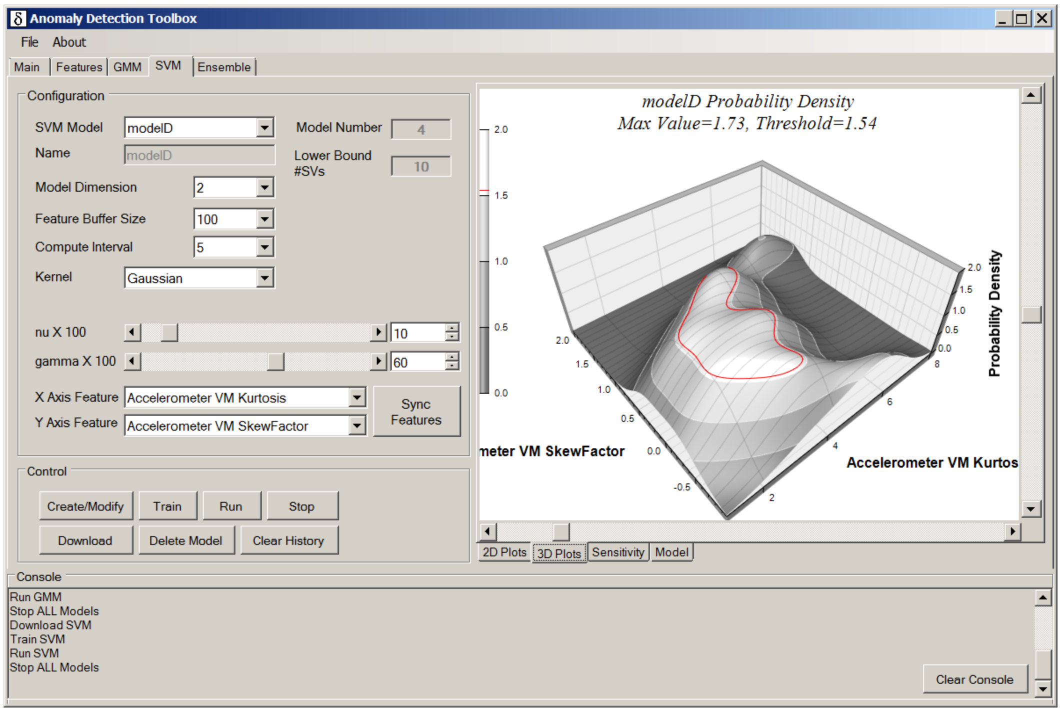
SVM Model: modelD
Name: modelD
Model Number: 4
Lower Bound #SVs: 10
Model Dimension: 2
Feature Buffer Size: 100
Compute Interval: 5
Kernel: Gaussian
nu X 100: 10
gamma X 100: 60
X Axis Feature: Accelerometer VM Kurtosis
Y Axis Feature: Accelerometer VM SkewFactor

```
Run GMM
Stop ALL Models
Download SVM
Train SVM
Run SVM
Stop ALL Models
```

Figure 19 SVM Screen – 2D Plots Tab

The 3D chart display is identical in function to that used for GMMs. The Sensitivity display appears the same, although the threshold is computed as part of the model training, and is not an input parameter to the model.

And of course, the Model display is specific to One Class Support Vector Machines.


Anomaly Detection Toolbox
File About
Main Features GMM SVM Ensemble
Create/Modify Train Run Stop
Download Delete Model Clear History
2D Plots 3D Plots Sensitivity Model
Clear Console
modelD Probability Density
Max Value=1.73, Threshold=1.54
Probability Density
meter VM SkewFactor
Accelerometer VM Kurtos

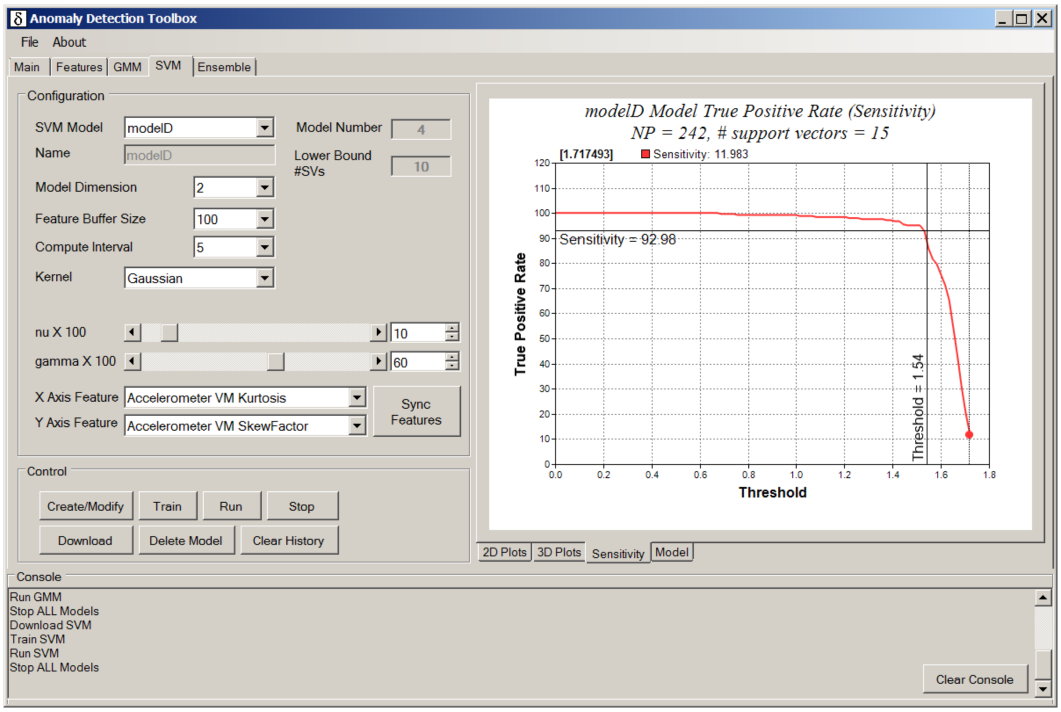
Configuration

SVM Model: modelD
Model Number: 4
Name: modelD
Lower Bound #SVs: 10
Model Dimension: 2
Feature Buffer Size: 100
Compute Interval: 5
Kernel: Gaussian
nu X 100: 10
gamma X 100: 60
X Axis Feature: Accelerometer VM Kurtosis
Y Axis Feature: Accelerometer VM SkewFactor
Sync Features

Console

```
Run GMM
Stop ALL Models
Download SVM
Train SVM
Run SVM
Stop ALL Models
```

Figure 20 SVM Screen – 3D Plots Tab


Anomaly Detection Toolbox
File About
Main Features GMM SVM Ensemble
Create/Modify Train Run Stop
Download Delete Model Clear History
2D Plots 3D Plots Sensitivity Model
Clear Console
modelD Model True Positive Rate (Sensitivity)
NP = 242, # support vectors = 15
[1.717493]
Sensitivity: 11.983
Sensitivity = 92.98
Threshold = 1.54
True Positive Rate
Threshold

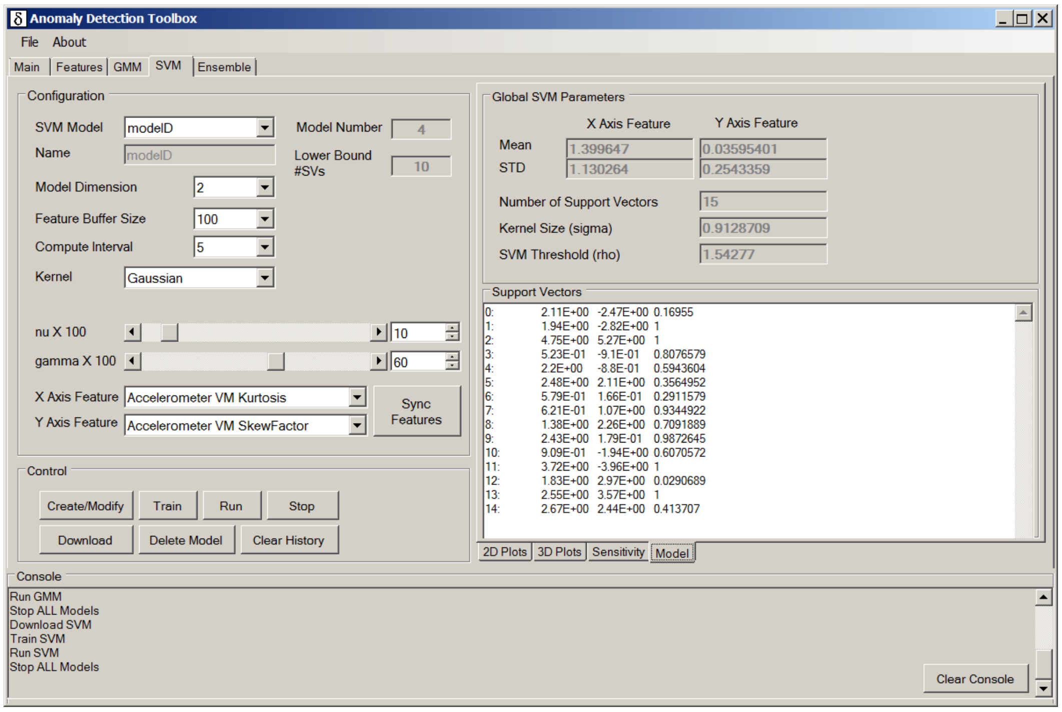
Configuration

SVM Model: modelD
Model Number: 4
Name: modelD
Lower Bound #SVs: 10
Model Dimension: 2
Feature Buffer Size: 100
Compute Interval: 5
Kernel: Gaussian
nu X 100: 10
gamma X 100: 60
X Axis Feature: Accelerometer VM Kurtosis
Y Axis Feature: Accelerometer VM SkewFactor
Sync Features

Console

```
Run GMM
Stop ALL Models
Download SVM
Train SVM
Run SVM
Stop ALL Models
```

Figure 21 SVM Screen – True Negative Rate Tab


Anomaly Detection Toolbox
File About
Main | Features | GMM | SVM | Ensemble
Sync Features
Create/Modify | Train | Run | Stop
Download | Delete Model | Clear History
2D Plots | 3D Plots | Sensitivity | Model
Clear Console

Configuration

SVM Model	modelD
Name	modelD
Model Number	4
Lower Bound #SVs	10
Model Dimension	2
Feature Buffer Size	100
Compute Interval	5
Kernel	Gaussian
nu X 100	10
gamma X 100	60
X Axis Feature	Accelerometer VM Kurtosis
Y Axis Feature	Accelerometer VM SkewFactor

Global SVM Parameters

	X Axis Feature	Y Axis Feature
Mean	1.399647	0.03595401
STD	1.130264	0.2543359

Number of Support Vectors	15
Kernel Size (sigma)	0.9128709
SVM Threshold (rho)	1.54277

Support Vectors

```
0:    2.11E+00  -2.47E+00  0.16955
1:    1.94E+00  -2.82E+00  1
2:    4.75E+00  5.27E+00   1
3:    5.23E-01  -9.1E-01   0.8076579
4:    2.2E+00   -8.8E-01   0.5943604
5:    2.48E+00  2.11E+00   0.3564952
6:    5.79E-01  1.66E-01   0.2911579
7:    6.21E-01  1.07E+00   0.9344922
8:    1.38E+00  2.26E+00   0.7091889
9:    2.43E+00  1.79E-01   0.9872645
10:   9.09E-01  -1.94E+00  0.6070572
11:   3.72E+00  -3.96E+00  1
12:   1.83E+00  2.97E+00   0.0290689
13:   2.55E+00  3.57E+00   1
14:   2.67E+00  2.44E+00   0.413707
```

Console

```
Run GMM
Stop ALL Models
Download SVM
Train SVM
Run SVM
Stop ALL Models
```

Figure 22 SVM Screen – Generated Model Tab

4.6 Ensemble Model Screen

4.6.1 Function

The ensemble model screen displays a table of all individual models computed to date. It can be used to select which of those models should be included in the ensemble model. It also specifies how many individual models must assert before the ensemble model asserts an anomaly.

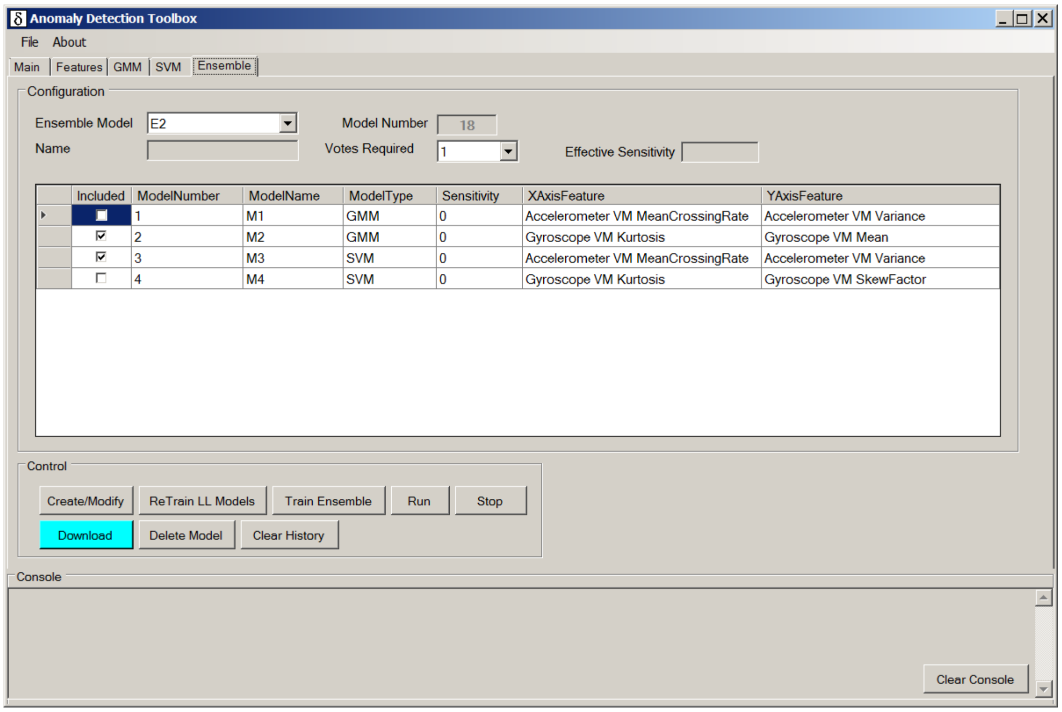
The Ensemble model screen includes:

- Configuration block, which includes controls for:
  - Ensemble Model: Select "new" to start defining a new model. Otherwise select from list of previously generated Ensemble Models.
  - Name: This field is read only unless the model selection is "new". In which case, use it to specify a name by which you will identify your model in the future.
  - Model Number: This is a read-only field. It contains a unique model number assigned once you click the "Create/Modify" button to save a model configuration.
  - Votes Required: Values are 1 to n, where n is the number of in-memory models.
  - A table of all lower level models created to date. You can individually choose to include or exclude each lower level model from your ensemble model. If you train the Ensemble model, then the checked set is used a starting point, with the option

to exclude any models which do not add value. If you do not train, then the set is used as-is.

- The Control block buttons have the same function as they do on GMM and SVM screen. However the Train button is split into:
  - Re-Train LL Models – forces all lower level models to be retrained over the same time period of feature samples.
  - Train Ensemble – runs a process on your pc[3] to select the optimal voting strategy and LL model inclusions, using the user defined values as a starting point.
- The Effective Sensitivity field displays an overall effective sensitivity rating for the ensemble model.

4.6.2 Screen Dump



Ensemble Model: E2

Name:

Model Number: 18

Votes Required: 1

Effective Sensitivity:

Included	ModelNumber	ModelName	ModelType	Sensitivity	XAxisFeature	YAxisFeature
☐	1	M1	GMM	0	Accelerometer VM MeanCrossingRate	Accelerometer VM Variance
✓	2	M2	GMM	0	Gyroscope VM Kurtosis	Gyroscope VM Mean
✓	3	M3	SVM	0	Accelerometer VM MeanCrossingRate	Accelerometer VM Variance
☐	4	M4	SVM	0	Gyroscope VM Kurtosis	Gyroscope VM SkewFactor

Figure 23 The Ensemble model tab

5. Matlab Interface

The File->Save->Matlab pulldown menu provides the capability to save:

- GMM model
- GMM features in .csv format
- SVM features in .csv format

[3] Lower level models are computed on the embedded board. Ensemble models are computed within the GUI and downloaded to the embedded board.

6. Symbols & Terms

Many of the following definitions have been pulled directly from outside references. We need to scrub and rephrase this section prior to external publication.

ANOVA	Analysis of Variance is a collection of statistical models used to analyze the differences between group means and their associated procedures
autocorrelation	the cross-correlation of a signal with itself. See http://en.wikipedia.org/wiki/Autocorrelation. $R(s,t) = \frac{\mathrm{E}[(X_t - \mu_t)(X_s - \mu_s)]}{\sigma_t \sigma_s}$,
CBM	Condition Based Monitoring
cepstrum	The power cepstrum of a signal is defined as the squared magnitude of the inverse Fourier transform of the logarithm of the squared magnitude of the Fourier transform of a signal. $= \left\vert \mathcal{F}^{-1}\left\{\log(\vert\mathcal{F}\left\{f(t)\right\}\vert^2)\right\}\right\vert^2$ The cepstrum can be seen as information about rate of change in the different spectrum bands.
CF	Crest Factor
concept drift	Refers to the notion that in some applications, statistics of the fielded system may move over time during normal operation. This implies the need for adaptive models.
correlation clustering	provides a method for clustering a set of objects into the optimum number of clusters without specifying that number in advance
crest factor	ratio of the peak values to the average value $CF = \frac{\max(\vert y(n)\vert)}{rms}$
dimension reduction	given a superset of features, dimension reduction encompasses various techniques used to identify only those features most important for classifying system states. Examples of dimension reduction techniques include Principal Component Analysis (PCA) and Singular Value Decomposition (SVD).
Entropy	a logarithmic measure of the number of states with significant probability of being occupied:

	$S = -k_{\mathrm{B}} \sum_i p_i \ln p_i \,,$ where k_B is the Bolzmann constant = 1.38065E-23 J/K. See http://en.wikipedia.org/wiki/Entropy for a full writeup.
feature	an alternate representation of sensor data. This may be FFT or wavelet coefficients, computed entropy, standard deviation, min/max, etc. Features are used as inputs to state detection and later stages in the ISO 13374 pipeline.
FFT	Fast Fourier Transform is an algorithm (there are many) to compute the discrete Fourier transform (DFT). $X_k = \sum_{n=0}^{N-1} x_n e^{-i 2 \pi k \frac{n}{N}} \qquad k = 0, \dots, N-1.$
ICA	Independent Component Analysis is a computational method for separating a multivariate signal into additive subcomponents. This is done by assuming that the subcomponents are non-Gaussian signals and that they are statistically independent from each other.
k-means clustering	a method of vector quantization, originally from signal processing, that is popular for cluster analysis in data mining. k-means clustering aims to partition nobservations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as aprototype of the cluster.
Kurtosis	a measure of the “peakedness” of the probability distribution of a real-valued random variable. This is the 4th central moment. $Kurtosis = \frac{\frac{1}{N}\sum_{n=1}^{N}[y(n)-\bar{y}]^4}{\sigma^4}$
MCM	Machine Condition Monitoring
naive Bayes classifier	a simple probabilistic classifier based on applying Bayes' theorem with strong (naive)independence assumptions.
PCA	Principal Component Analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components
PHM	Prognostics and Health Management

PSD	Power Spectral Density describes how the power of a signal or time series is distributed over the different frequencies $$S_{xx}(\omega) = \frac{(\Delta t)^2}{T} \left\| \sum_{n=1}^{N} x_n e^{-i\omega n} \right\|^2$$
RMS	Root Mean Square, defined as: $$rms = \sqrt{\frac{\sum_{n=1}^{N} y(n)^2}{N}}$$
segmentation	the process of breaking a continuous stream of signal values into a series of “buckets” or “time slots” for further analysis (ex: FFT).
SF	Shape Factor
shape factor	a measure of the compactness of a signal $$SF = \frac{rms}{\overline{y}}$$
SK	Skew Factor
skew factor	the normalized 3rd central moment: $$SK = \frac{\frac{1}{N}\sum_{n=1}^{N} [\, y(n) - \overline{y}\,]^3}{\sigma^3}$$
standard deviation	the square root of the variance, AKA STD $$\begin{aligned} \sigma &= \sqrt{\mathrm{E}[(X-\mu)^2]} \\ &= \sqrt{\mathrm{E}[X^2] + \mathrm{E}[(-2\mu X)] + \mathrm{E}[\mu^2]} = \sqrt{\mathrm{E}[X^2] - 2\mu\,\mathrm{E}[X] + \mu^2} \\ &= \sqrt{\mathrm{E}[X^2] - 2\mu^2 + \mu^2} = \sqrt{\mathrm{E}[X^2] - \mu^2} \\ &= \sqrt{\mathrm{E}[X^2] - (\mathrm{E}[X])^2} \end{aligned}$$
SVD	Singular Value Decomposition is a technique where a matrix M is factored into UΣV* where: U is a mxm unitary matrix

	Σ is an mxn rectangular diagonal matrix
	V* is an nxn unitary matrix
	SVD can be used as a technique for dimension reduction.
SVM	Support vector machines are supervised learning models with associated learning algorithms that analyze data and recognize patterns, used for classification and regression analysis.
variance	$\sigma^2 = E[X^2] – E[X]^2$

$$\operatorname{Var}(X) = \sigma^2 = \int (x-\mu)^2 f(x)\,dx = \int x^2 f(x)\,dx - \mu^2$$

where f(x) is the probability density function for X and

$$\mu = \int x\, f(x)\, dx$$

7. References

The following references were used during development of this project:

Chris Aldrich and Lida Auret, “Unsupervised Process Monitoring and Fault Diagnosis with Machine Learning Methods”, Springer-Verlag, 2013

Ralf Herbrich, “Learning Kernel Classifiers”, The MIT Press, 2002

Yaser S. Abu-Mostafa, Malik Magdon-Ismail and Husan-Tien Lin, “Learning from Data”, AMLbook.com, 2012

“Vibration Diagnostic Guide”, SKF Reliability Systems

“Harris Shock and Vibration Handbook, 6th edition”, Allan G. Piersol and Thomas L. Paez, McGraw Hill

J. Wang, J. Lee and C. Zhang, "Kernel Trick Embedded Gaussian Mixture Model", ResearchGate, 2003. [Online]. Available: https://www.researchgate.net/publication/221394219_Kernel_Trick_Embedded_Gaussian_Mixture_Model. [Accessed: 02- Sep- 2016].

Olivier Schwander and Frank Nielsen, “Learning mixtures by simplifying kernel density estimators”

Matej Kristan, Danijel Skocaj and Ales Leonardis, “Incremental learning with Gaussian mixture models”.

Luan Soares Oliveira, Gustavo E.A.P.A. Batista, “IGMM-CD: A Gaussian Mixture Classification Algorithm for Data Streams with Concept Drifts”, 2015 Brazilian Conference on Intelligent Systems

Marco Pimentel, Lei Clifton, David Clifton and L Tarassenko, “A review of novelty detection”, Elsevier, Signal Processing 99 (2014), pages 215-249

Hodge, V.J. and Austin, J, (2004) “A survey of outlier detection methodologies”, Artificial Intelligence Review, 22, pp 85-126

Manuel Davy, Frederic Desobry, Arthur Gretton and Christian Doncarli, “An online support vector machine for abnormal events detection”, Elsevier, 2005

Mennatallah Amer, Markus Goldstein and Slim Abdennadher, “Enhancing One-class Support Vector Machines for Unsupervised Anomaly Detection”, ODD’13, 2013

Paul Bodesheim, et. al., “Kernel Null Space Methods for Novelty Detection”, Proc. CVPR 2013, pp 3374-3381

Markos Markou and Sameer Singh, “Novelty detection: a review-part 1: statistical approaches”, Signal Processing 83 (2003), pp 2481-2497

Markos Markou and Sameer Singh, “Novelty detection: a review-part 2: neural network based approaches”, Signal Processing 83 (2003), pp 2499-2521

Shiblee Sadik and Le Gruenwald, “Online Outlier Detection for Data Streams”, IDEAS '11 Proceedings of the 15th Symposium on International Database Engineering & Applications, 2011

Karanjit Singh and Dr. Shuchita Upadhyaya, “Outlier Detection: Applications and Techniques”, IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 1, No. 3, January 2012

Ludmila I. Kuncheva and William J. Faithfull, “PCA Feature Extraction for Change Detection in Multidimensional Unlabelled Streaming Data”, 21st International Conference on Pattern Recognition, November 11-15, 2012.

Andrei Bara, “DeADA Self-adaptive anomaly detection dataflow architecture”, MEng Individual Project, Department of Computing, Imperial College London, June 2013.

Chih-Chung Chang and Chih-Jen Lin, LIBSVM : a library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2:27:1--27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsv

8. Serial Communications Interface

8.1 Summary

Table 2. Summarizes the various command packet formats for the Anomaly Detection Toolbox. Excluding the discovery command, frame tags for response packets are the

same, except that bit 7 is set to indicate SUCCESS. Bit 7 remains clear if the command did not execute properly.

Table 2. Embedded Command Summary

Command	**R/W**	**Frame Tag**	
		Send / Receive (Error)	**Receive (Success)**
Device Info		0x60	
Download Features	W	0x25	0xA5
Reserved for Set FFT Feature Configuration (Φ2 Feature)	W	0x26	0xA6
Set GMM Configuration	W	0x28	0xA8
Set SVM Configuration	W	0x29	0xA9
Set Model Mode[1]	W	0x2A	0xAA
Read Sample Rates	R	0x2B	0xAB
Set Ensemble Configuration	W	0x2C	0xAC
Read GMM Model	R	N/A	0x41
Read SVM Model	R	N/A	0x42
Model Result Data Packet	R	N/A	0x40

[1] Mode options = Delete / Train / Apply

8.2 HDLC Encoding

Development board to Anomaly Detection Toolbox communication uses a streaming data protocol. Packets are delimited by inserting a special byte (0x7E) between packets. This means that we must provide a means for transmitting 0x7E within the packet payload. This is done on the transmission side by making the following substitutions:

- Replace 0x7E by 0x7D 0x5E (1 byte payload becomes 2 transmitted)
- Replace 0x7D by 0x7D 0x5D (1 byte payload becomes 2 transmitted)

The Toolbox does the inverse mapping as the data stream is received. Partial packets are discarded. The main tab of the toolbox has a “View Raw Serial Traffic” option available if you wish to see the raw byte stream.

8.3 Endianess

The 2 byte ISSDK payload length field in all packets is in Big Endian (MSB first) form. All other 16 and 32-bit quantities are in Little Endian (LSB first) format.

Strings are arrays characters (bytes), and therefore the first character of a string is located at the first byte, then second, …

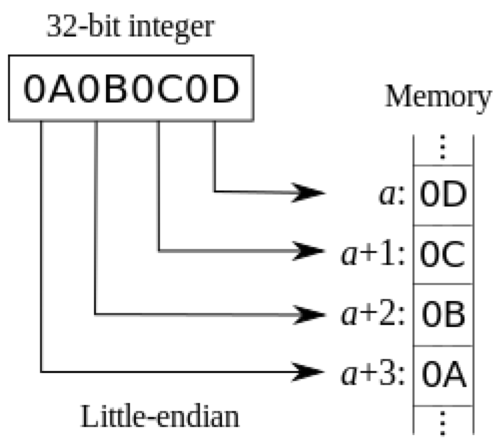


Figure 24 Little Endian Byte Addressing

8.3.1 IEEE 754-2008 32-bit floating point format

Logically, 32-bit IEEE floating point values have the sign bit in bit 31 (upper left below) and the F0 in bit 0 (lower right below)

Table 3. IEEE 754-2008 Single Precision Floating Point

Floating point format							
S	EXP7	EXP6	EXP5	EXP4	EXP3	EXP2	EXP1
EXP0	F22	F21	F20	F19	F18	F17	F16
F15	F14	F13	F12	F11	F10	F9	F8
F7	F6	F5	F4	F3	F2	F1	F0

Floating points components are:

- S = Sign bit
- E[7:0] = Exponent
- F[22:0] = Fractional Field

The Little Endian convention is that the LSB has the lowest **byte** number. So the quantity above is actually visualized in memory and in the serial command/response stream as:

Table 4. Little Endian Floating Point Encoding

offset	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
0	F7	F6	F5	F4	F3	F2	F1	F0

1	F15	F14	F13	F12	F11	F10	F9	F8
2	EXP0	F22	F21	F20	F19	F18	F17	F16
3	S	EXP7	EXP6	EXP5	EXP4	EXP3	EXP2	EXP1

8.4 Device Discovery

This screen uses the ISSDK Device Info Command.

8.4.1 Command Packet

The frame tag has value 0x60 for this command.

Table 5. Device Discovery Command Packet

Field Name	Num Bytes	Individual Bits							
		7	6	5	4	3	2	1	0
Frame Tag	1	Reserved = 0x0	Interface ID = 0x3		Reserved = 0x00				

8.4.2 Response Packet

Table 6. Device Discovery Response Packet

Field Name	Num Bytes	Individual Bits							
		7	6	5	4	3	2	1	0
Frame Tag	1	0x60							
Host Protocol Version	1	Currently 0x10							
Application Name Length	1	AppNameLen							
Application Name	AppNameLen	Name characters							
Board Name Length	1	BoardNameLen							
Board Name	BoardNameLen	Name characters							
Shield Name Length	1	ShieldNameLen							
Shield Name	ShieldNameLen	Name characters							

8.5 Features Enable/Disable

8.5.1 Command Packet

At startup, all features are disabled by default. Previously stored values may be loaded using the “Restore” button on the Main tab. Once any value has changed, the “Download to Device” button is highlighted to indicate that the embedded board is out of sync with the GUI. Clicks the “Download to Device” button to update the embedded device with current settings. Upon successful receipt of an acknowledgment packet, the button highlight is cleared. The following command structure is associated with that button event:

Table 7. Feature Enable/Disable Command Packet

Field Name	Num Bytes	Hex Values	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Frame Tag =	1	0x25	0x0	Interface ID = 0x1		Command Type = 0x5 (enable/disable features)				
Sequence ID	1	0x00	Varies, assigned by GUI as unique command identifier							
Length	2	0x0016	Payload Size (uint16, MSB first)							
Accel X Features	2		Bitfield for enabled features (see Table 8.)							
Accel Y Features	2		Bitfield for enabled features (see Table 8.)							
Accel Z Features	2		Bitfield for enabled features (see Table 8.)							
Accel VM Features	2		Bitfield for enabled features (see Table 8.)							
Gyro X Features	2		Bitfield for enabled features (see Table 8.)							
Gyro Y Features	2		Bitfield for enabled features (see Table 8.)							
Gyro Z Features	2		Bitfield for enabled features (see Table 8.)							
Temperature Features	2		Bitfield for enabled features (see Table 8.)							
Pressure Features	2		Bitfield for enabled features (see Table 8.)							
Microphone Features	2		Bitfield for enabled features (see Table 8.)							
FeatureInterval	4		Interval over which features are computed in ms (100 minimum, 500 default)							

Table 8. Feature Type Bit-Field Encoding

Sensor Type	Encoding
Mean	0x0040
Variance	0x0020
Skew Factor	0x0010
Kurtosis	0x0008
Mean Crossing Rate	0x0004
STD	0x0002
STD(X/Y/Z)/STD(VM)	0x0001

8.5.2 Response Packet

Response packets only indicate success/failure, as shown in Table 9.

Table 9. Feature Enable/Disable Response Packet

Field Name	Num Bytes	Hex Value	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Frame Tag	1	0x25 or 0xA5	Status	Interface ID = 0x1		Command Type = 0x5 (enable/disable features)				
Sequence ID	1	0x00	Varies, assigned by GUI as unique command identifier							
Length	2	0x0000								

The Status bit in the Frame Tag field is set to 0x1 if the command executed successfully, and 0x0 otherwise.

8.6 Read Sample Rates

8.6.1 Command Packet

Sample rates are fixed by the embedded application, and are automatically retrieved anytime you switch to the “Features” tab. The following command/response packets are used for that communication.

Table 10. Read Sample Rates Command Packet

Field Name	Num Bytes	Hex Values	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Frame Tag	1	0x2B	0x0	Interface ID = 0x1		Command Type = 0xB				
Sequence ID	1	0x00	Varies, assigned by GUI as unique command identifier							
Length	2	0x0001	1 byte dummy payload							
Payload	1	0x00	Dummy							

8.6.2 Response Packet

Table 11. Read Sample Rates Response Packet

Field Name	Num Bytes	Hex Values	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Frame Tag	1	0x2B or 0xAB	Status	Interface ID = 0x1		Command Type = 0xB				
Sequence ID	1	0x00	Varies, assigned by GUI as unique command identifier							
Length	2	0x14	Payload length = 0x0014 (decimal 20, MSB first)							
AccSR	4		Accelerometer sample rate in uint32 format (Little Endian)							
GyroSR	4		Gyroscope sample rate in uint32 format (Little Endian)							
PressureSR	4		Pressure sensor sample rate in uint32 format (Little Endian)							
TempSR	4		Temperature sensor sample rate in uint32 format (Little Endian)							
MicSR	4		Microphone sample rate in uint32 format (Little Endian)							

8.7 Set GMM Configuration

This command can be used to create a new GMM instance, or to modify an existing one.

8.7.1 Command Packet

Table 12. Set GMM Configuration Command Packet

Field Name	Num Bytes	Hex Values	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Frame Tag	1	0x28	0x0	Interface ID = 0x1		Command Type = 0x8 (set GMM model configuation)				
Sequence ID	1	Varies	Varies, assigned by GUI as unique command identifier							
Length	2	0x000A or 0x000C	Payload Size = decimal 10 for one dimensional models, decimal 12 for two dimensional models. Format = uint16 format (MSB first).							
Model Type	1	Varies	Kernel Type – see Table 21. “No Kernel” is the only currently supported option				Model Type - see Table 14.			
Model Number	1	Varies	RES	RES	Unique model ID for this specific model instance					
Control	1	Varies	RES	RES	RES	DIM	Maximum Number of Gaussians			
BufSize	2	Varies	Feature Buffer Size (uint16)							
MCInterval	2	Varies	Model Compute Interval (uint16)							
ThresholdX100	1	Varies	0x00 to 0x64 (unsigned)							
F1A	1	Varies	Feature Code for Feature 1 – see Table 17.							
F1B	1	Varies	Sensor Code for Feature 1 – see Table 15.				Axis Code For Feature 1 - See Table 16.			
F2A	1	Varies	Feature Code for Feature 2 - see Table 17. F2A and F2B are omitted for one dimensional models							
F2B	1	Varies	Sensor Code for Feature 2 – see Table 15.				Axis Code For Feature 1 - See Table 16.			

Table 13. Individual Field Descriptions

Field Name	Description
DIM	0: Model has one feature input 1: Model has two feature inputs
Maximum Number of Gaussians	Range: 1-7 This is the maximum number of Gaussians that the model will attempt to fix. The final model may have fewer Gaussians.

RES	Reserved bit field

Table 14. Model Type Bit-Field Encoding

Model Type	Encoding
GMM	0x1
SVM	0x2
Ensemble	0x3
All others	Reserved

Table 15. Sensor Encoding

Sensor Type	Encoding
Accelerometer	0x0
Gyroscope	0x1
Magnetometer	0x2
Temperature	0x3
Pressure	0x4
Microphone	0x5

Table 16. Axis Encoding

Sensor Type	Encoding
X	0x0
Y	0x1
Z	0x2
Vector Magnitude	0x3
Scalar	0x4

Table 17. Feature typedef Encoding

Sensor Type	Encoding
Mean	0x00000000
Variance	0x00000001
Skew Factor	0x00000010
Kurtosis	0x00000011
Mean Crossing Rate	0x00000100
STD	0x00000101
STD(X/Y/Z)/STD(VM)	0x00000110
Crest Factor	0x00000111
Reserved	All others

8.7.2 Response Packet

The response packet format is shown in Table 9.

Table 18. Set GMM Configuration Response Packet

Field Name	Num Bytes	Hex Value	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Frame Tag	1	0x28or 0xA8	Status	Interface ID = 0x1		Command Type = 0x8				
Sequence ID	1	0x00	Varies, assigned by GUI as unique command identifier							
Length	2	0x0001								
ErrorCode	1	See Table 19.								

Table 19. Model Mode Error Conditions

Code	Error Condition
0x00	No Error
0x01	Modify failed because model not found
0x02	Create failed because model does not exist
0x03	Feature1 not enabled (not applicable to Ensemble models)
0x04	Feature2 not enabled (not applicable to Ensemble models)
0x05	Sub-Model not found (Ensemble models only)
All others	Reserved

8.8 Set SVM Configuration

This command can be used to create a new SVM instance, or to modify an existing one.

8.8.1 Command Packet

Table 20. Set SVM Configuration Command Packet

Field Name	Num Bytes	Hex Values	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Frame Tag	1	0x29	0x0	Interface ID = 0x1		Command Type = 0x9 (set SVM model configuation)				
Sequence ID	1	Varies	Varies, assigned by GUI as unique command identifier							
Length	2	0x000C or 0x000E	Payload Size = decimal 12 for one dimensional models, decimal 14 for two dimensional models. Format = uint16 format (MSB first).							
Model Type	1	Varies	Kernel Type – see Table 21.				Model Type - see Table 14.			
Model Number	1	Varies	RES	RES	Unique model ID for this specific model instance					
Control	1	Varies	DIM	RESERVED						
BufSize	2	Varies	Feature Buffer Size (uint16)							
MCInterval	2	Varies	Model Compute Interval (uint16)							
nu X 100	1	Varies	0x00 to 0x64 (unsigned)							
gamma X 100	1	Varies	100 X gamma value to use in SVM computations. Sigma = 1/sqrt(gamma)							
F1A	1	Varies	Feature Code for Feature 1 - see Table 17.							
F1B	1	Varies	Sensor Code for Feature 1 – see Table 15.				Axis Code For Feature 1 - See Table 16.			
F2A	1	Varies	Feature Code for Feature 2 - see Table 17. F2A and F2B are omitted for one dimensional models							
F2B	1	Varies	Sensor Code for Feature 2 – see Table 15.				Axis Code For Feature 1 - See Table 16.			

Table 21. Kernel Encodings

Code	Kernel Type
0x0	No Kernel
0x1	Gaussian
0x1-0x7	Reserved

Table 22. Individual Field Descriptions

Field Name	Description
DIM	0: Model has one feature input 1: Model has two feature inputs
RES	Reserved bit field

8.8.2 Response Packet

The response packet format is shown in Table 9.

Table 23. Set GMM Configuration Response Packet

Field Name	Num Bytes	Hex Value	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Frame Tag	1	0x29 or 0xA9	Status	Interface ID = 0x1		Command Type = 0x9				
Sequence ID	1	0x00	Varies, assigned by GUI as unique command identifier							
Length	2	0x0001								
ErrorCode	1	See Table 19.								

8.9 Set Ensemble Configuration

This command can be used to create a new Ensemble instance, or to modify an existing one.

8.9.1 Command Packet

Table 24. Set Ensemble Configuration Command Packet

Field Name	Num Bytes	Hex Values	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Frame Tag	1	0x2C	0x0	Interface ID = 0x1		Command Type = 0xC (set Ensemble model configuration)				
Sequence ID	1	Varies	Varies, assigned by GUI as unique command identifier							
Length	2	2 + #models	Payload Size. Format = uint16 format (MSB first).							
Model Type	1	Varies	Number of Sub-Models				0x3 = Model Type - see Table 14.			
Model Number	1	0x11	RES	RES	Unique Ensemble Model ID					
Control	1	Varies	RES	RES	RES	RES	Votes Required			
Sub-Model Specifiers	1		1st model number							
	1		2nd model number							
	…		…							
	1		nth model number							

8.9.2 Response Packet

The response packet format is shown in Table 9.

Table 25. Set GMM Configuration Response Packet

Field Name	Num Bytes	Hex Value	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Frame Tag	1	0x2C or 0xAC	Status	Interface ID = 0x1		Command Type = 0xC				
Sequence ID	1	0x00	Varies, assigned by GUI as unique command identifier							
Length	2	0x0001								
ErrorCode	1	See Table 19.								

8.10 Set Model Mode

8.10.1 Command Packet

This command is used to control model operations for existing models

Table 26. Set Model Mode Command Packet

Field Name	Num Bytes	Hex Values	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Frame Tag	1	0x2A	0x0	Interface ID = 0x1		Command Type = 0xA (set model mode)				
Sequence ID	1	Varies	Varies, assigned by GUI as unique command identifier							
Length	2	0x0003	Payload Size							
Model Type	1	Varies	RES	RES	RES	RES	Model Type (see Table 14.)			
Model Number	1	Varies	RES	RES	Unique model ID for this specific model instance					
Control	1	Varies	Stream Enable	TED	Delete All	Delete	Train	Run	Stop	CLR

Table 27. Individual Field Descriptions

Field Name	Description
Stream Enable	0: Do nothing 1: stream features and model results
TED	Train Ensemble Dependencies 0: Do nothing 1: start retraining all models upon which the given ensemble is dependent (only applicable to ensemble models)
Delete All	0: Do nothing 1: Delete all models
Delete	0: Do nothing 1: Delete this model
Train	0: Do nothing 1: Train this model. Not applicable to Ensemble models. Training for those is done within the GUI.
Run	0: Do nothing 1: Run this model
Stop	0: Do nothing 1: Stop all training and run functions. Disable all models.
CLR	0: Do nothing 1: Clear ALL embedded feature FIFOs (not just those applicable to this model). If this bit is asserted concurrently with another, this function should be implemented first.

Response Packet

The response packet format is shown in Table 28.

Table 28. Set Model Mode Response Packet

Field Name	Num Bytes	Hex Value	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Frame Tag	1	0x2A or 0xAA	Status	Interface ID = 0x1		Command Type = 0xA (set model mode)				
Sequence ID	1	0x00	Varies, assigned by GUI as unique command identifier							
Length	2	0x0000	Zero Payload Length							

8.11 Model Result Packet

The model result packet uses the ISSDK isochronous command structure. Packets are initiated by the embedded device and there is no response packet by the GUI. Packets are only sent when the stream enable bit in the model mode command packet has been set.

Table 29. Model Result Data Packet Format

Field Name	Num Bytes	Hex Values	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Frame Tag	1	0x40	0x0	Interface ID = 0x2		0x00				
Length	2	0x0007 OR 0x000B	Payload Size = 3 for ensembles, 5 for single feature models, 9 for two-feature models							
Model Number	1	Varies	P/F	T/R	Unique model ID for this specific model instance					
Feature Interval	2	Varies	Feature Interval Number in uint16 format (Little Endian)							
Feature 1[2]	4	Varies	IEEE 754-2008 32-bit floating point format							
Feature 2[1, 2]	4	Varies	IEEE 754-2008 32-bit floating point format							

1. present, but not applicable for single feature models
2. not present in ensemble models

Feature 1 and 2 are in IEEE 754-2008 32-bit floating point format. Values are raw feature values computed PRIOR to any scaling done to condition the values for machine learning.

P/F = Pass (1) / Fail (0)

Feature 1 and Feature 2 match the feature definitions in the model configuration commands which defined those models.

Note that if multiple models use the same input feature, it will be transmitted multiple times.

The Feature Interval Number is used to identify the unique feature epoch for each result packet. The first epoch starts at “0”, and it increments by 1 with each new epoch. All model result packets sent during a given epoch will share the same value. The value wraps at 16 bits.

8.12 GMM Computed Model Packet

The model result packet uses the ISSDK isochronous command structure. Packets are initiated by the embedded device and there is no response packet by the GUI. Packets are only sent when the stream enable bit in the model mode command packet has been set.

Table 30. GMM Computed Model Packet Format

Field Name	Num Bytes	Hex Values	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Frame Tag	1	0x41	0x0	Interface ID = 0x2		0x01				
Length	2	Varies	Total Payload Size includes space for model number, feature interval, NumGaussians and parameters for each of the Gaussians.							
Model Number	1	Varies	P/F	T/R	Unique model ID for this specific model instance					
Feature Interval	2	Varies	Feature Interval Number in uint16 format (Little Endian)							
Control	1	0x00 or 0x01	RES	RES	RES	RES	RES	RES	RES	DIM
Feature 1 Mean	4	Varies	IEEE 754-2008 32-bit floating point format							
Feature 2 Mean[1]	4	Varies	IEEE 754-2008 32-bit floating point format							
Feature 1 STD	4	Varies	IEEE 754-2008 32-bit floating point format							
Feature 2 STD[1]	4	Varies	IEEE 754-2008 32-bit floating point format							
NumGaussians	1	Varies	Number of Gaussian components Note: The following block is repeated once per Gaussian component.							
muMeans[0]	4	Varies	IEEE 754-2008 32-bit floating point format							
muMeans[1][1]	4	Varies	IEEE 754-2008 32-bit floating point format							
covariance[0,0]	4	Varies	IEEE 754-2008 32-bit floating point format							
Covariance[0,1][1]	4	Varies	IEEE 754-2008 32-bit floating point format							
Covariance[1,0][1]	4	Varies	IEEE 754-2008 32-bit floating point format							
Covariance[1,1][1]	4	Varies	IEEE 754-2008 32-bit floating point format							
pi (mixing probability)	4	Varies	IEEE 754-2008 32-bit floating point format							

1. not applicable for single feature models

8.13 OC-SVM Computed Model Packet

The model result packet uses the ISSDK isochronous command structure. Packets are initiated by the embedded device and there is no response packet by the GUI. Packets are only sent when the stream enable bit in the model mode command packet has been set.

Table 31. Model Result Data Packet Format

Field Name	Num Bytes	Hex Values	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Frame Tag	1	0x42	0x0	Interface ID = 0x2		0x02				
Length	2	Varies	Total Payload Size includes space for model number, feature interval, NumSVs, coefficients, SVs, and rho (threshold) trained.							
Model Number	1	Varies	P/F	T/R	Unique model ID for this specific model instance					
Feature Interval	2	Varies	Feature Interval Number in uint16 format (Little Endian)							
Control	1	0x00 or 0x01	RES	RES	RES	RES	RES	RES	RES	DIM
Feature 1 Mean	4	Varies	IEEE 754-2008 32-bit floating point format							
Feature 2 Mean[1]	4	Varies	IEEE 754-2008 32-bit floating point format							
Feature 1 STD	4	Varies	IEEE 754-2008 32-bit floating point format							
Feature 2 STD[1]	4	Varies	IEEE 754-2008 32-bit floating point format							
rho	4	Varies	IEEE 754-2008 32-bit floating point format the computed threshold by training							
NumSVs	1	Varies	Number of Support Vectors (SVs) Note: The following block is repeated once per SV.							
nu	4	Varies	IEEE 754-2008 32-bit floating point format							
kernelSize	4	Varies	IEEE 754-2008 32-bit floating point format							
coefficent	4	Varies	IEEE 754-2008 32-bit floating point format							
SV[0]	4	Varies	IEEE 754-2008 32-bit floating point format							
SV[1] [1]	4	Varies	IEEE 754-2008 32-bit floating point format							

1. not applicable for single feature models

8.14 Ensemble Computed Model Packet

The computed model packet for Ensemble models has exactly the same form as the “Set Ensemble Configuration” packet defined in Section 8.9. There is no response packet sent from the GUI back to the embedded board as a result of receiving this packet.

al information

9.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

9.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the *Terms and conditions of commercial sale* of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Translations — A non-English (translated) version of a document is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Evaluation products — This product is provided on an "as is" and "with all faults" basis for evaluation purposes only. NXP Semiconductors, its affiliates and their suppliers expressly disclaim all warranties, whether express, implied or statutory, including but not limited to the implied warranties of non-infringement, merchantability and fitness for a particular purpose. The entire risk as to the quality, or arising out of the use or performance, of this product remains with customer.

In no event shall NXP Semiconductors, its affiliates or their suppliers be liable to customer for any special, indirect, consequential, punitive or incidental damages (including without limitation damages for loss of business, business interruption, loss of use, loss of data or information, and the like) arising out the use of or inability to use the product, whether or not based on tort (including negligence), strict liability, breach of contract, breach of warranty or any other theory, even if advised of the possibility of such damages.

Notwithstanding any damages that customer might incur for any reason whatsoever (including without limitation, all damages referenced above and all direct or general damages), the entire liability of NXP Semiconductors, its affiliates and their suppliers and customer's exclusive remedy for all of the foregoing shall be limited to actual damages incurred by customer based on reasonable reliance up to the greater of the amount actually paid by customer for the product or five dollars (US$5.00). The foregoing limitations, exclusions and disclaimers shall apply to the maximum extent permitted by applicable law, even if any remedy fails of its essential purpose.

9.3 Licenses

Purchase of NXP <xxx> components
<License statement text>

9.4 Patents

Notice is herewith given that the subject device uses one or more of the following patents and that each of these patents may have corresponding patents in other jurisdictions.

<Patent ID> — owned by <Company name>

9.5 Trademarks

Notice: All referenced brands, product names, service names and trademarks are property of their respective owners.

<Name> — is a trademark of NXP Semiconductors N.V.

10. List of figures

11. List of tables

12. Contents

Date of release: 17 January 2017

Document identifier: NXPUserManual_v2.11.0A_1

Date of release: 17 January 2017

Document identifier: NXPUserManual_v2.11.0A_1