Machine Learning from MCU to Cloud – i.MX RT Platform for ML

Michael Stanley

System Integrator

MICR Security & IoT Solutions Team

June 2019 | Session #AMF-EDG-T3643

SECURE CONNECTIONS FOR A SMARTER WORLD

Course Focus

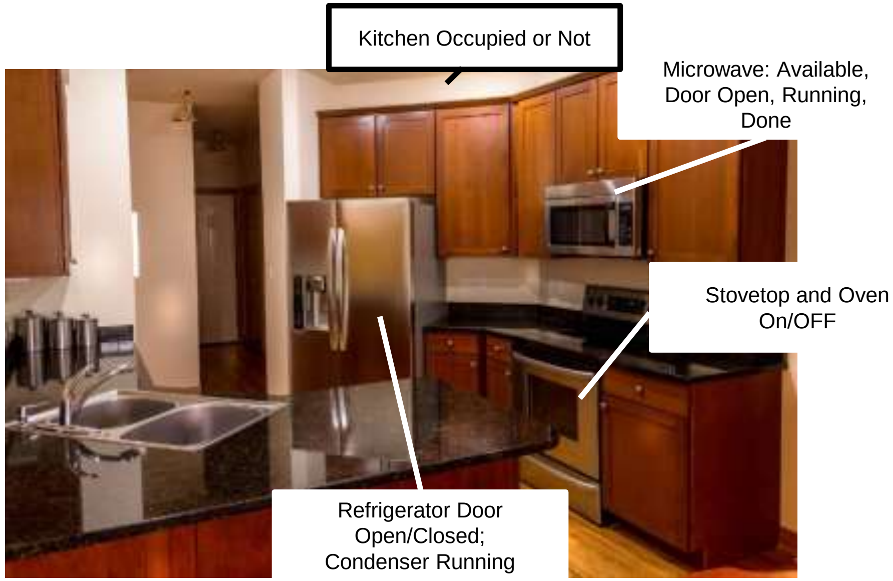
- Examine a hardware/software solution enabling machine learning for machine, home & building condition monitoring.
- Optimizing your time to market
- 1st peak at cloud interfaces using Microsoft Azure

Agenda

- Solution Overview
- Hardware Components
- Firmware Components
- Data Collection & Off-Line Training
- Reality AI as an Option
- Microsoft Azure Interfaces
- Availability & Contacts

Target Applications

- White Goods / Smart Home
- Environmental Monitoring
- Retrofit motor control solutions to include vibration/thermal analysis to detect stall & load imbalance
- Home Security Systems / Smart Locks



Kitchen Occupied or Not

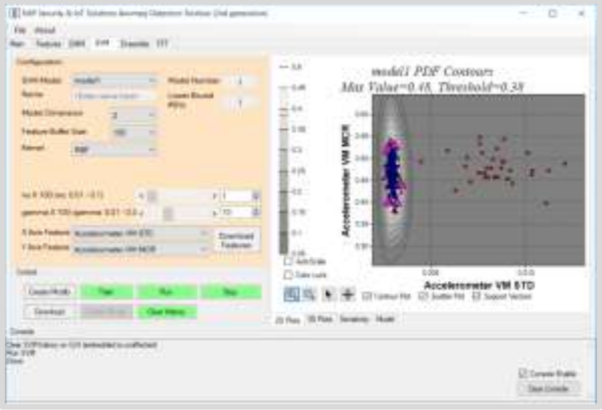
Microwave: Available, Door Open, Running, Done

Stovetop and Oven On/OFF

Refrigerator Door Open/Closed; Condenser Running

Solution Overview

Off the shelf hardware for sensor data collection and product prototyping.



Windows-based tool for data collection for machine learning and measuring application metrics

Reality AI

AI at the Edge

Expert 3rd party SaaS models can be directly integrated into the framework.

Embedded firmware engine for machine learning delivered as a custom SDK

SDK

jupyter

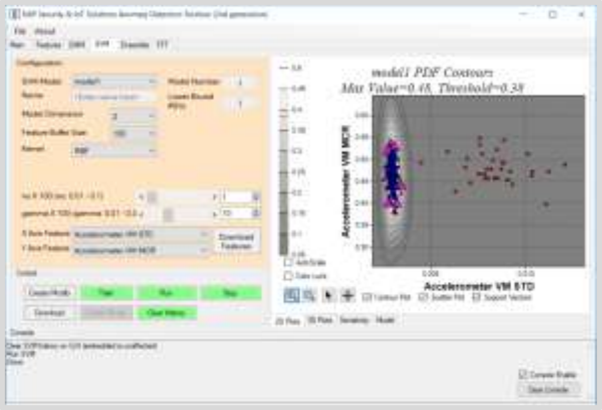
OpenCV

python

Easily integrate models trained via standard open source tools.

Solution Overview

Off the shelf hardware for sensor data collection and product prototyping.



Windows-based tool for data collection for machine learning and measuring application metrics

Reality AI®

AI at the Edge

Expert 3rd party SaaS models can be directly integrated into the framework.

Embedded firmware engine for machine learning delivered as a custom SDK

SDK

Coming soon…

Microsoft Azure

Take advantage of Azure cloud infrastructure and ML

jupyter

OpenCV

python

Easily integrate models trained via standard open source tools.

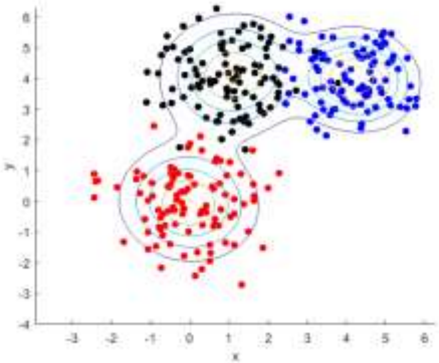
This Solution Leverages Traditional ML Techniques… Not Deep Neural Networks

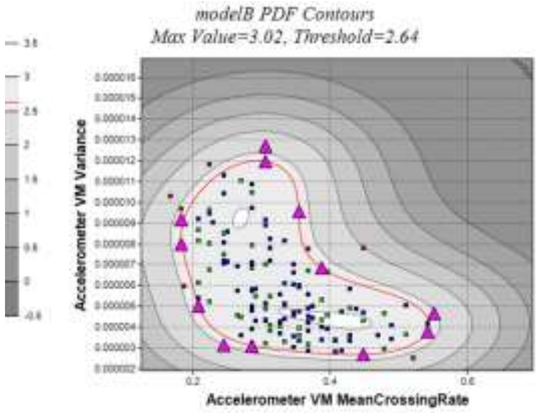
- Focus is on simpler sensor types
- Less data is required for training
- Utilize one-class algorithms when possible for anomaly detection
- More intuitive models
- Smaller models mean:
  - less power
  - models can be implemented on lower tier MCUs
- In situ training is feasible (for one class problems), which in term enables adaptive models

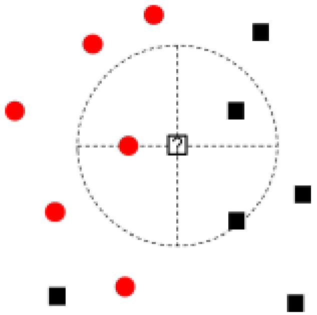
Supported Model Types

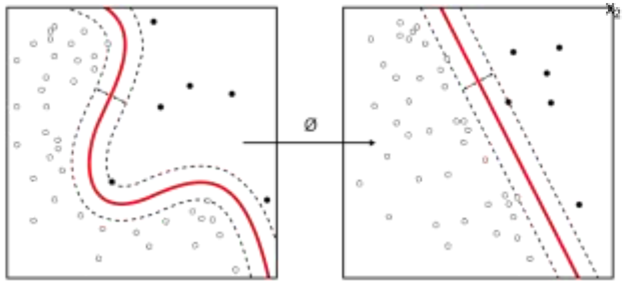
OpenCV model types derived from the StatModel base class:

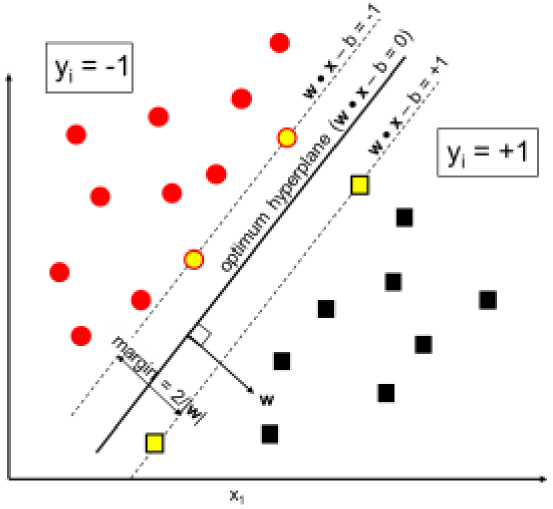
- Naïve Bayes
- Expectation Maximization
- Decision Trees
- Random Trees
- Multi-layer Perceptrons
- Logistic Regression
- Support Vector Machines
- K-Nearest Neighbors
- Boosted Trees

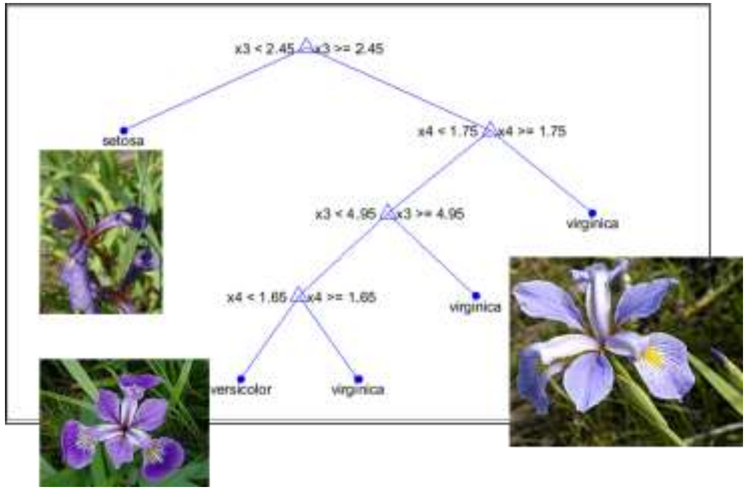












What Does the Customer Get from Us?

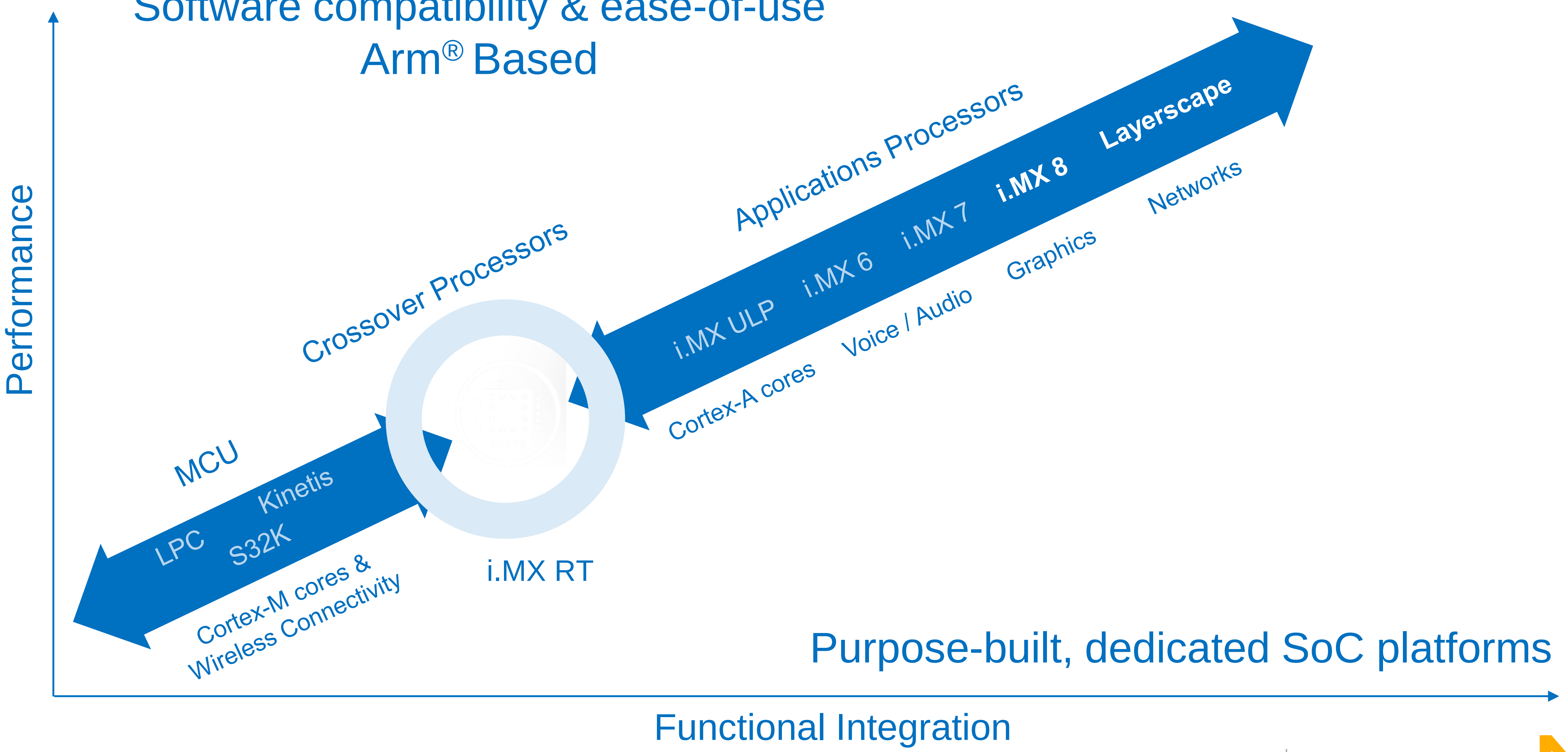
- i.MXRT106x base board / sensor board stack-up
- Production ready C++ framework that includes:
  - Very simple programming interface at the customer’s level
  - Machine Learning Engine that handles
    - Sensor sampling
    - Automated feature extraction
    - Run computed model(s) – OpenCV or partner-generated
    - Train new OpenCV models model(s)
- 2nd Generation GUI for data collection and viewing model results
- Jupyter notebooks for off-line feature and model engineering
- Full documentation and CAD files for hardware and software

Hardware Components

Software presented in this discussion is bundled with purchase of the i.MX RT106C

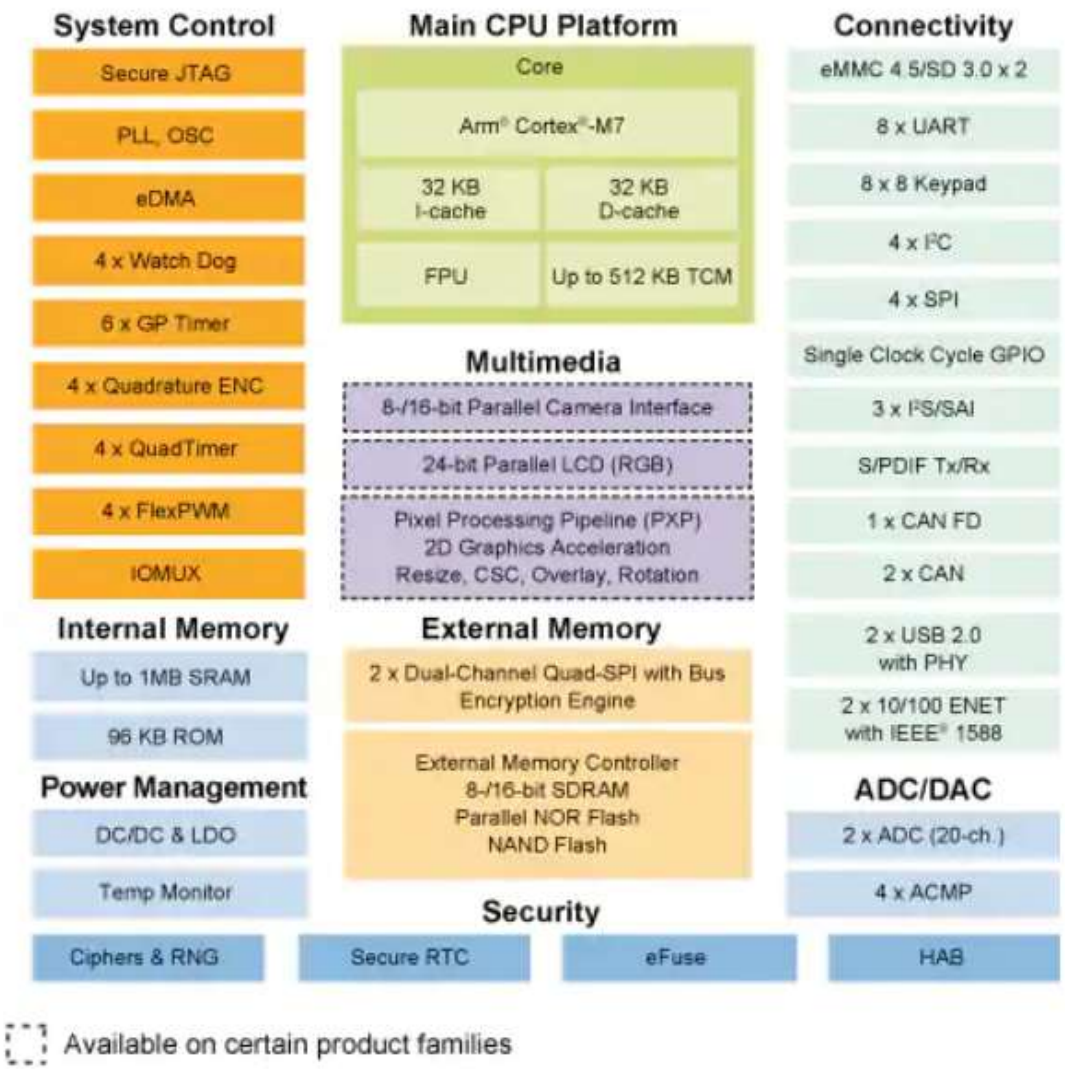
Hybrid MCU

NXP Scalable Processing Continuum



i.MX RT106C

- Solution-specific variant of the i.MX RT1060 family targeting embedded machine learning applications
- ARM® Cortex®-M7 Operating at up to 600MHz
- 1MB internal RAM plus external memory interface
- Numerous communications options
- Low cost / high performance ideally matched for classic machine learning algorithms and applications



System Control

Secure JTAG

PLL, OSC

eDMA

4 x Watch Dog

6 x GP Timer

4 x Quadrature ENC

4 x QuadTimer

4 x FlexPWM

IOMUX

Internal Memory

Up to 1MB SRAM

96 KB ROM

Power Management

DC/DC & LDO

Temp Monitor

Main CPU Platform

Core

Arm® Cortex®-M7

32 KB I-cache

32 KB D-cache

FPU

Up to 512 KB TCM

Multimedia

8-/16-bit Parallel Camera Interface

24-bit Parallel LCD (RGB)

Pixel Processing Pipeline (PXP)
2D Graphics Acceleration
Resize, CSC, Overlay, Rotation

External Memory

2 x Dual-Channel Quad-SPI with Bus Encryption Engine

External Memory Controller
8-/16-bit SDRAM
Parallel NOR Flash
NAND Flash

Connectivity

eMMC 4.5/SD 3.0 x 2

8 x UART

8 x 8 Keypad

4 x I^2C

4 x SPI

Single Clock Cycle GPIO

3 x I^2S/SAI

S/PDIF Tx/Rx

1 x CAN FD

2 x CAN

2 x USB 2.0 with PHY

2 x 10/100 ENET with IEEE® 1588

ADC/DAC

2 x ADC (20-ch.)

4 x ACMP

Security

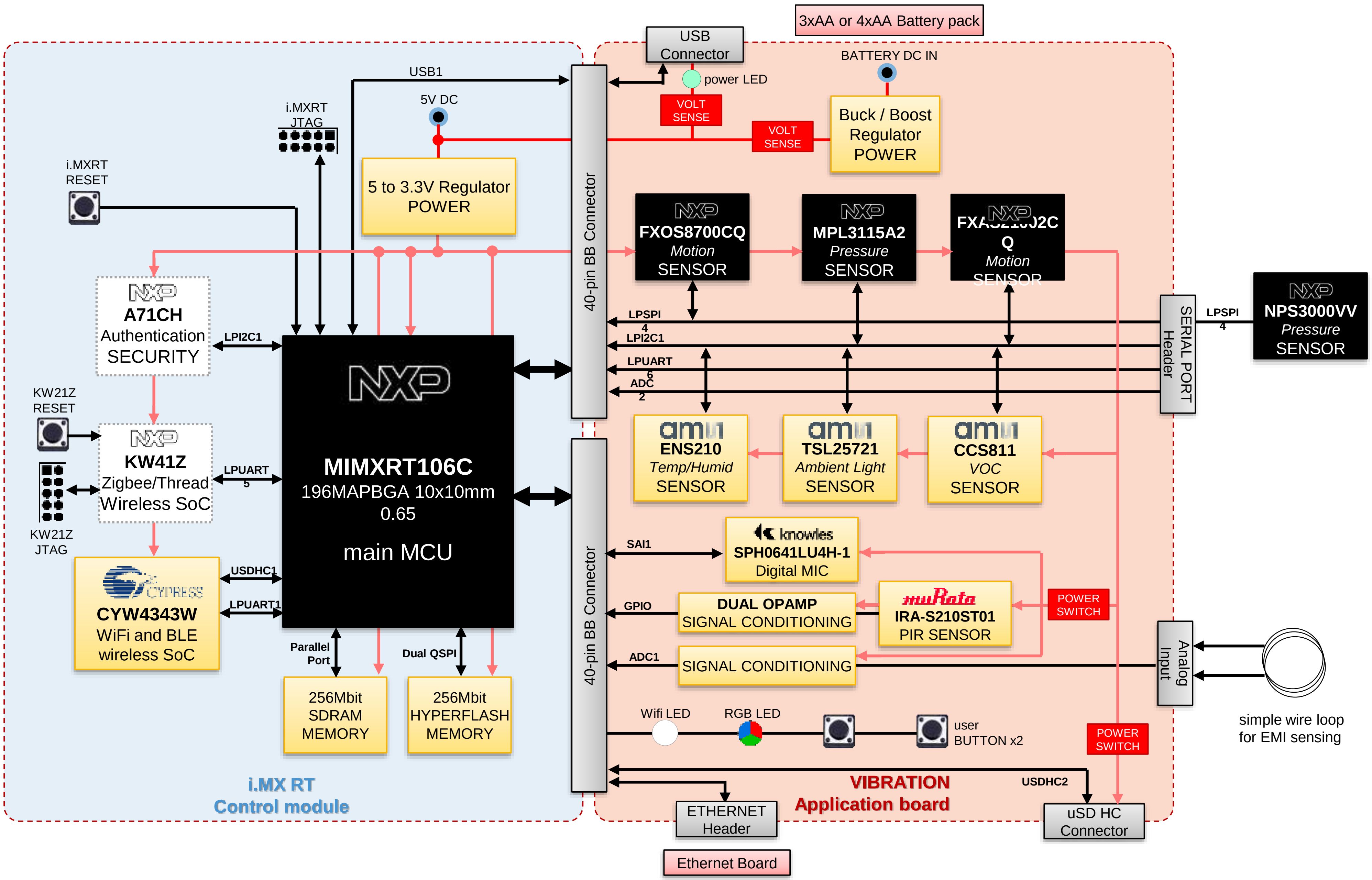
Ciphers & RNG

Secure RTC

eFuse

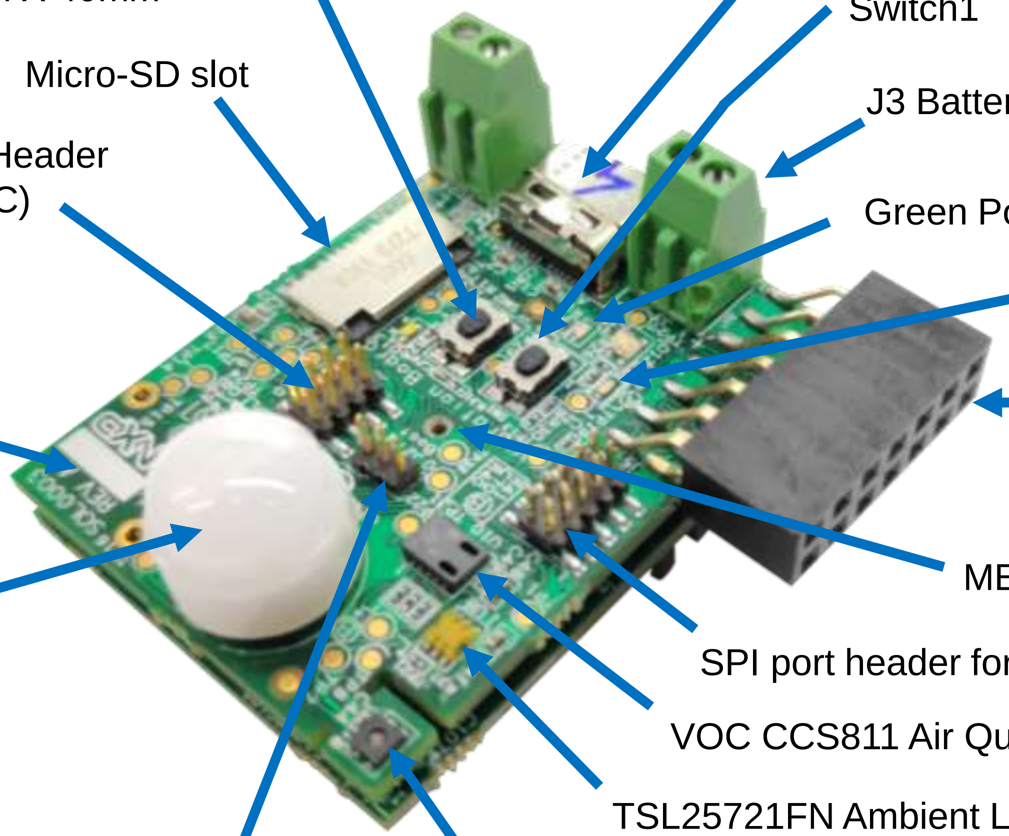
HAB

Available on certain product families


3xAA or 4xAA Battery pack
USB Connector
BATTERY DC IN
power LED
VOLT SENSE
VOLT SENSE
Buck / Boost Regulator POWER
USB1
5V DC
i.MXRT JTAG
i.MXRT RESET
5 to 3.3V Regulator POWER
40-pin BB Connector
FXOS8700CQ Motion SENSOR
MPL3115A2 Pressure SENSOR
FXAS21002C Q Motion SENSOR
A71CH Authentication SECURITY
LPI2C1
LPSPI 4
LPI2C1
LPUART 6
ADC 2
LPSPI 4
SERIAL PORT Header
NPS3000VV Pressure SENSOR
KW21Z RESET
KW41Z Zigbee/Thread Wireless SoC
LPUART 5
KW21Z JTAG
MIMXRT106C
196MAPBGA 10x10mm
0.65
main MCU
ENS210 Temp/Humid SENSOR
TSL25721 Ambient Light SENSOR
CCS811 VOC SENSOR
CYW4343W WiFi and BLE wireless SoC
USDHC1
LPUART1
SAI1
knowles SPH0641LU4H-1 Digital MIC
GPIO
DUAL OPAMP SIGNAL CONDITIONING
IRA-S210ST01 PIR SENSOR
POWER SWITCH
ADC1
SIGNAL CONDITIONING
Analog Input
simple wire loop for EMI sensing
Parallel Port
Dual QSPI
256Mbit SDRAM MEMORY
256Mbit HYPERFLASH MEMORY
40-pin BB Connector
Wifi LED
RGB LED
user BUTTON x2
POWER SWITCH
USDHC2
i.MX RT Control module
VIBRATION Application board
ETHERNET Header
uSD HC Connector
Ethernet Board

2nd Generation Sensor Hardware

Board dimensions: 30mm X 40mm



J10 EMI Sensor Connector

Switch2

USB Type C Connector

Switch1

The FXAS2100 gyro and MPL3115A2 absolute pressure sensor are on the bottom of the sensor board

Micro-SD slot

J3 Battery Connector

Serial Port Header (UART & I2C)

Green Power LED

RGB LED (D12)

FXOS8300CQ mag/accel is on the bottom of the sensor board

J5 Ethernet Header (only applicable for wired Ethernet use)

MEMS Mic Port

PIR Sensor

SPI port header for NPS3000VV differential pressure sensor

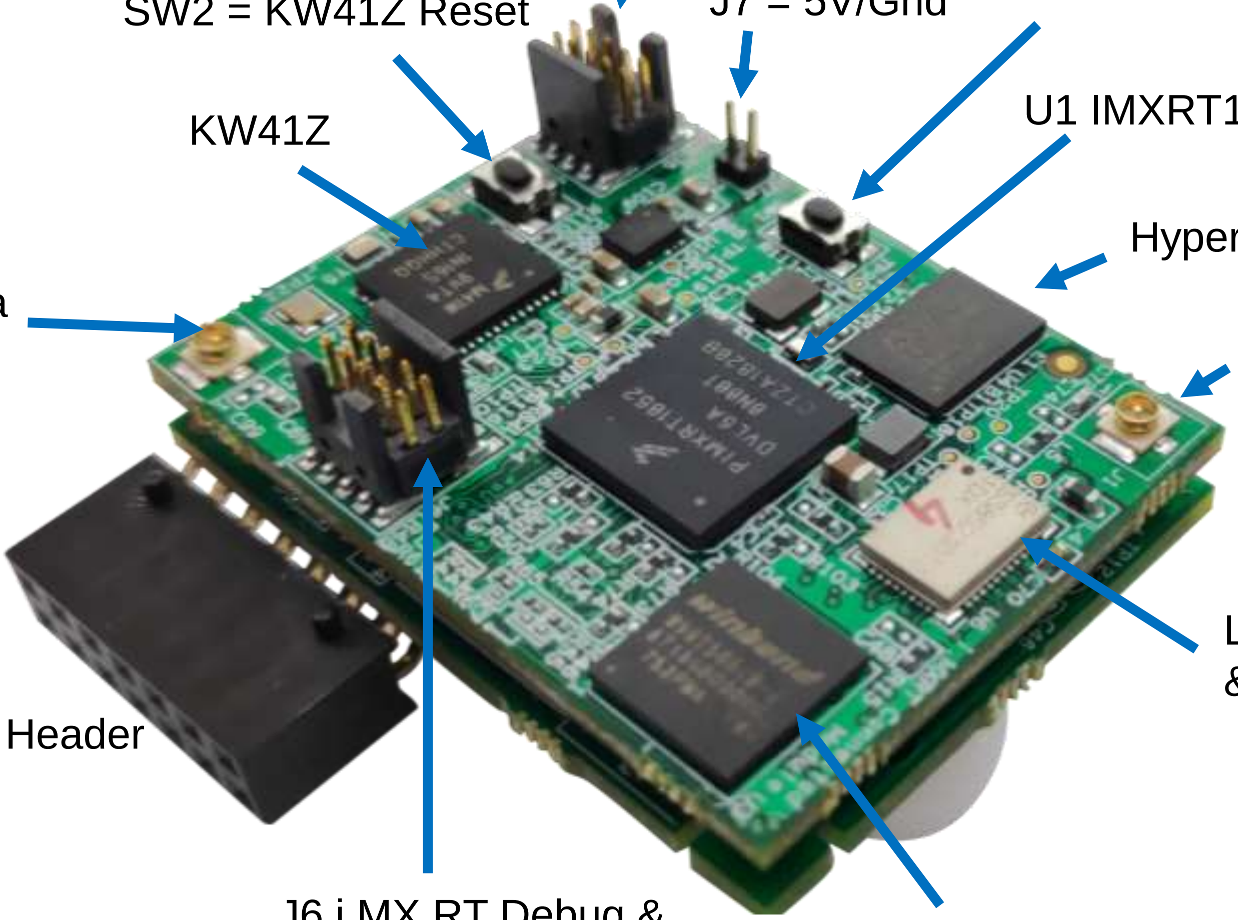
VOC CCS811 Air Quality Sensor

TSL25721FN Ambient Light Sensor

J27 Boot Mode Selection

ENS210 Temp/Humidity Sensor

MCU Board



J5 = KW41Z Debug & Programming Header

SW2 = KW41Z Reset

J7 = 5V/Gnd

SW1 = i.MXRT Reset

U1 IMXRT106x MCU (ARM® Cortex® M7)

KW41Z

HyperFlash

KW41 RF Antenna

LBEE5KL1DX-833 Antenna

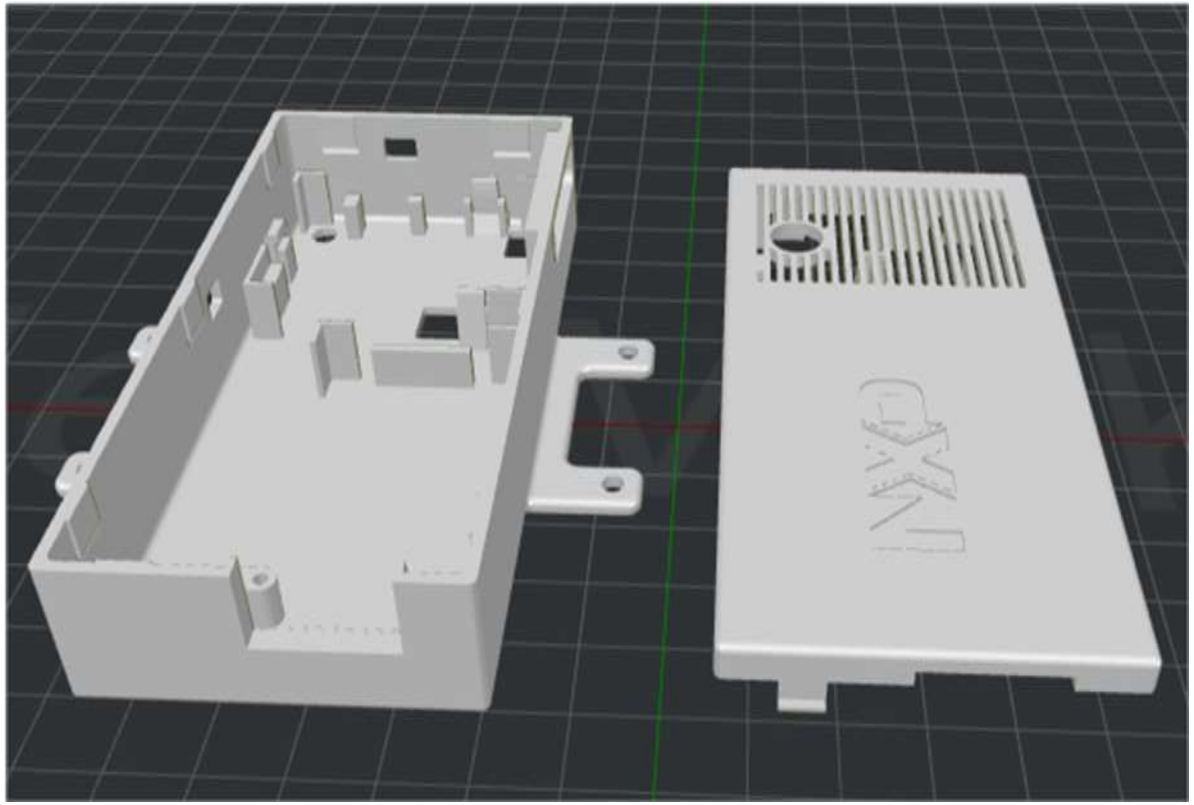
LBEE5KL1DX-833 WiFi & BT/BLE Module

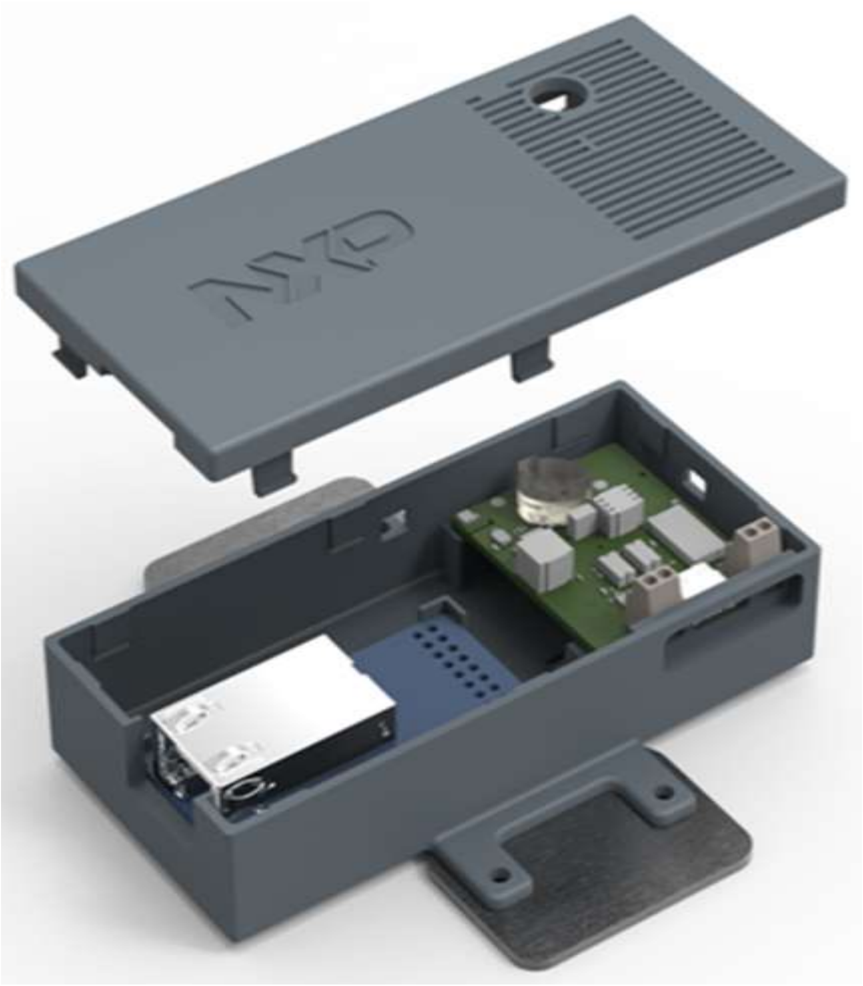
J5 Ethernet Header

J6 i.MX RT Debug & Programming Header

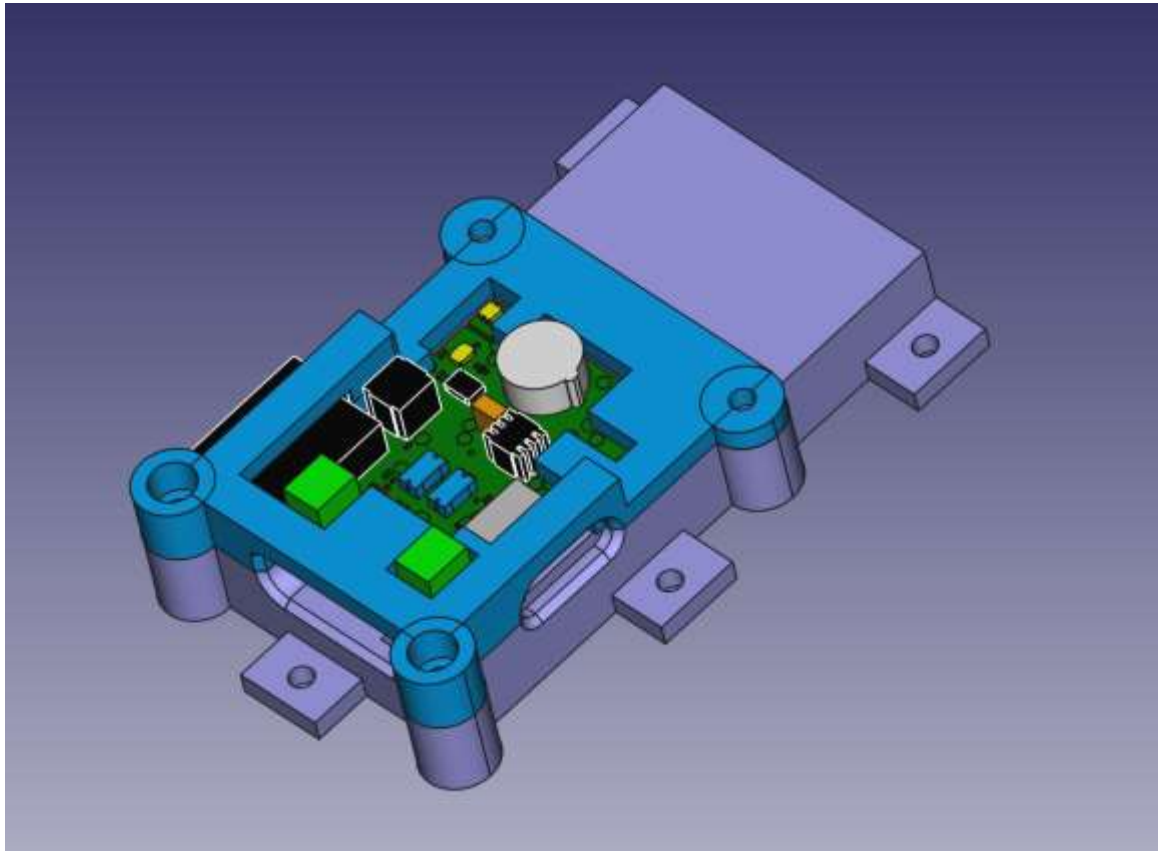
SDRAM

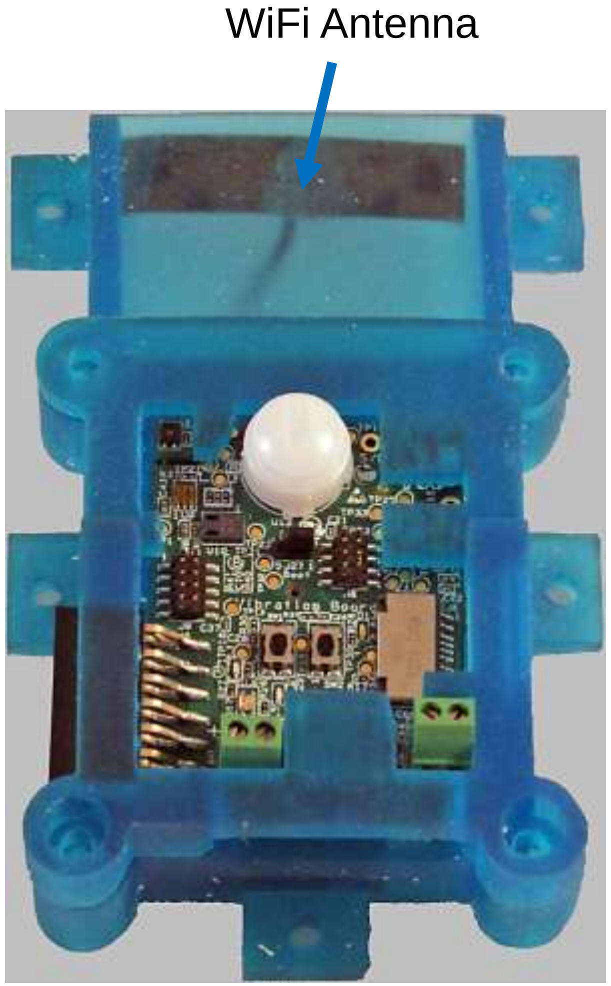
MCU/Sensor Board + Ethernet Adapter Enclosure





MCU/Sensor Board Only Enclosure With WIFI



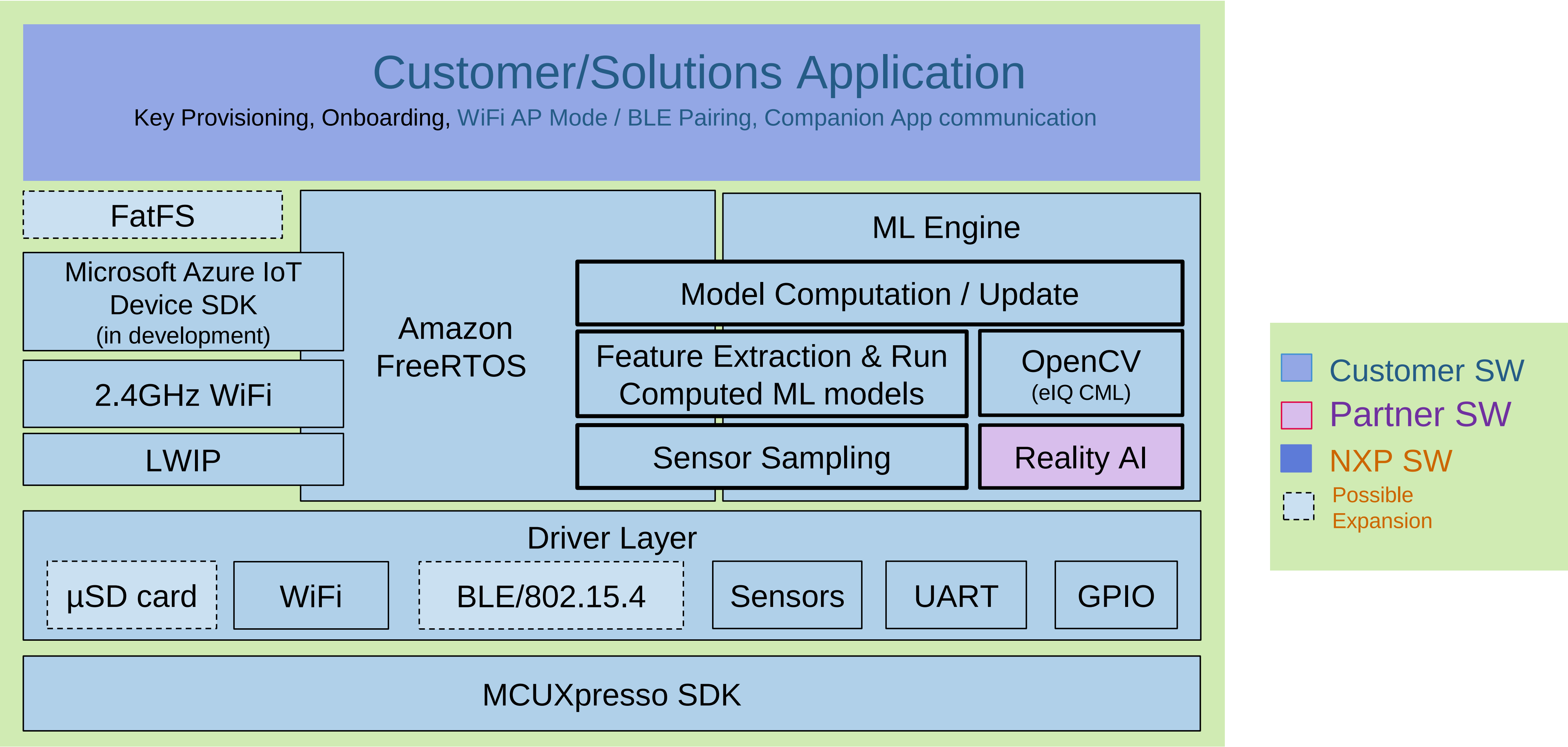


Firmware Components

The firmware is delivered in the form of a custom MCUXpresso SDK:

Software presented in this discussion is bundled with purchase of the i.MX RT106C Hybrid MCU

Solution Definition – High Level Software Block Diagram



Customer/Solutions Application

Key Provisioning, Onboarding, WiFi AP Mode / BLE Pairing, Companion App communication

FatFS

Microsoft Azure IoT Device SDK (in development)

2.4GHz WiFi

LWIP

Amazon FreeRTOS

ML Engine

Model Computation / Update

Feature Extraction & Run Computed ML models

OpenCV (eIQ CML)

Sensor Sampling

Reality AI

Driver Layer

µSD card

WiFi

BLE/802.15.4

Sensors

UART

GPIO

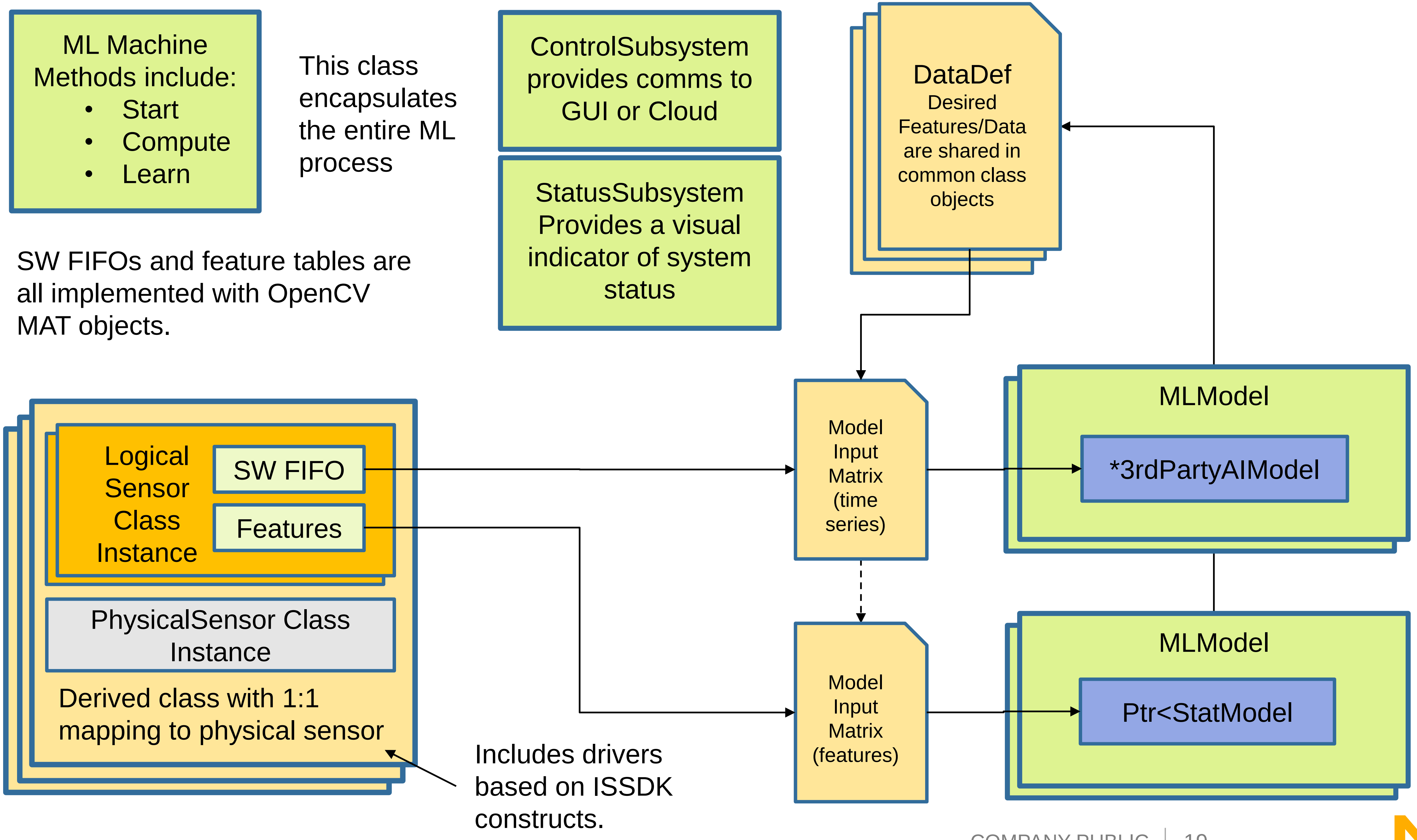
MCUXpresso SDK

🟦 Customer SW

🟪 Partner SW

🟦 NXP SW

⬚ Possible Expansion



ML Machine Methods include:

- Start
- Compute
- Learn

This class encapsulates the entire ML process

ControlSubsystem provides comms to GUI or Cloud

StatusSubsystem Provides a visual indicator of system status

DataDef

Desired Features/Data are shared in common class objects

SW FIFOs and feature tables are all implemented with OpenCV MAT objects.

Logical Sensor Class Instance

SW FIFO

Features

PhysicalSensor Class Instance

Derived class with 1:1 mapping to physical sensor

Includes drivers based on ISSDK constructs.

Model Input Matrix (time series)

Model Input Matrix (features)

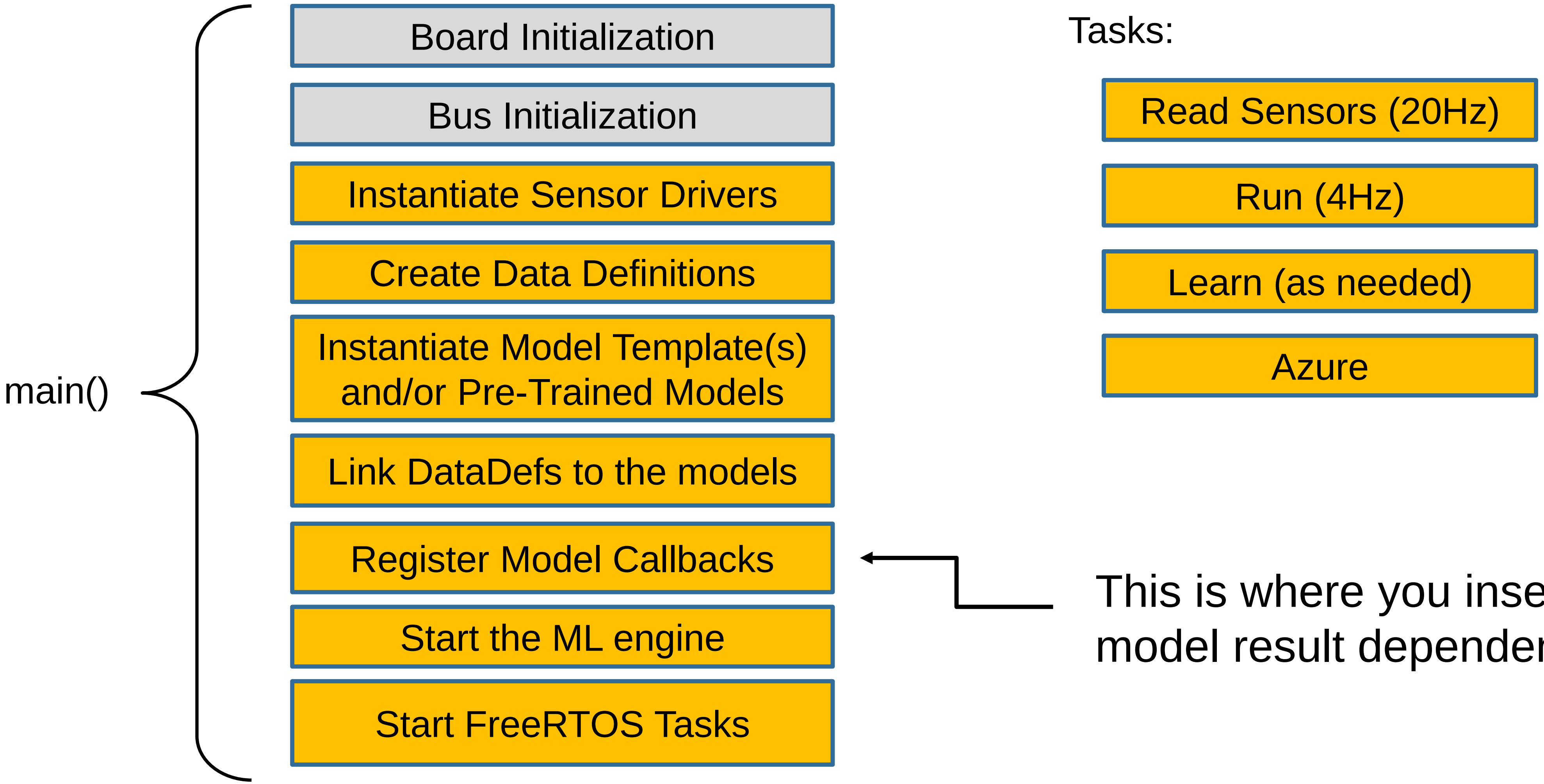
MLModel

*3rdPartyAIModel

MLModel

Ptr<StatModel

Major Components of the Embedded Program Flow



main()

- Board Initialization
- Bus Initialization
- Instantiate Sensor Drivers
- Create Data Definitions
- Instantiate Model Template(s) and/or Pre-Trained Models
- Link DataDefs to the models
- Register Model Callbacks
- Start the ML engine
- Start FreeRTOS Tasks

Tasks:

- Read Sensors (20Hz)
- Run (4Hz)
- Learn (as needed)
- Azure

This is where you insert your model result dependent code

Example Model Wrapper From Python

```
tuple<MLModelPtr_t, Ptr<RTrees>> add_custom_trained_model0(uint8_t modelNumber, MLMachinePtr_t mlMachine,
         string modelName, string FXAS, string FXOS){
 //The user is responsible for instantiating the required sensors at the top level and passing their
identifiers to this function

 Mat featureMeans = Mat(1, 4, CV_32F, model0_means);
 Mat featureStds = Mat(1, 4, CV_32F, model0_stds);

 Ptr<DataDef> f0 = mlMachine->addDataDef(FXOS,"Accel",AXIS_X,FEATURE_CORRVY);
 Ptr<DataDef> f1 = mlMachine->addDataDef(FXOS,"Accel",AXIS_X,FEATURE_CORRVZ);
 Ptr<DataDef> f2 = mlMachine->addDataDef(FXOS,"Accel",AXIS_VM,FEATURE_VARIANCE);
 Ptr<DataDef> f3 = mlMachine->addDataDef(FXAS,"Gyroscope",AXIS_Y,FEATURE_MEAN);

 Ptr<RTrees> rtm = StatModel::loadFromString<RTrees>(modelXMLFile0);
 MLModelPtr_t RTM = mlMachine->addModelTemplate(modelNumber, modelName, MODEL_TYPE_OpenCV_rtrees, rtm);
 RTM->setMeanStddev(&featureMeans, &featureStds);

 RTM->addInput(f0);
 RTM->addInput(f1);
 RTM->addInput(f2);
 RTM->addInput(f3);

 RTM->setMode(MODEL_MODE_TRAINED);
 RTM->lock();
 return(std::make_tuple(RTM, rtm));
}
```

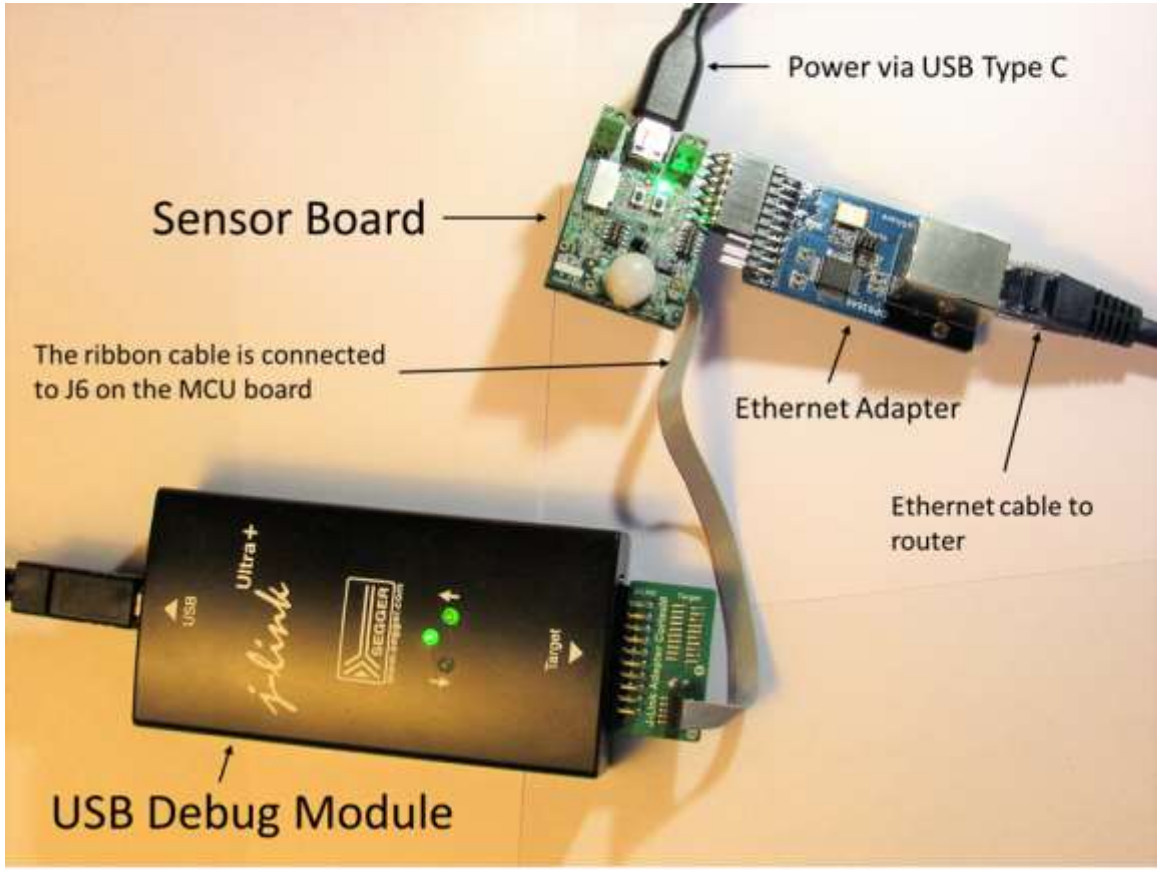
Example Code Snippet From Main()

```
 // Instantiate FXOS8700CQ sensor driver for 3 axis accelerometer + 3 axis magnetometer
PhysicalSensorPtr_t fxos8700cq = new FXOS8700CQ(0, 1, SPIdrv, &spiFxosDeviceInfo, true, true, 200);
mlMachine->installSensor(fxos8700cq, "FXOS");

// Instantiate FXAS21002CQ 3-axis gyroscope sensor driver
PhysicalSensorPtr_t fxas21002cq = new FXAS21002CQ(0x20, 1, I2Cdrv, &i2cDevInfo, true, true, 200);
mlMachine->installSensor(fxas21002cq, "FXAS");

MLModelPtr_t M0; // Wrapper object for the model
Ptr<RTrees> m0;  // OpenCV model pointer
tie(M0, m0) = add_custom_trained_model0(++modelsCreated, mlMachine, "m0", "FXAS", "FXOS");
M0->registerCallback(model_callback0);
M0->startRunning();
```

Development Environment



- i.MXRT106x MCU + Sensor Board
- 3rd party Ethernet adapter or WIFI
- Segger Jlink debug module recommended
- Free MCUXpresso IDE

Data Collection & Off-Line Training

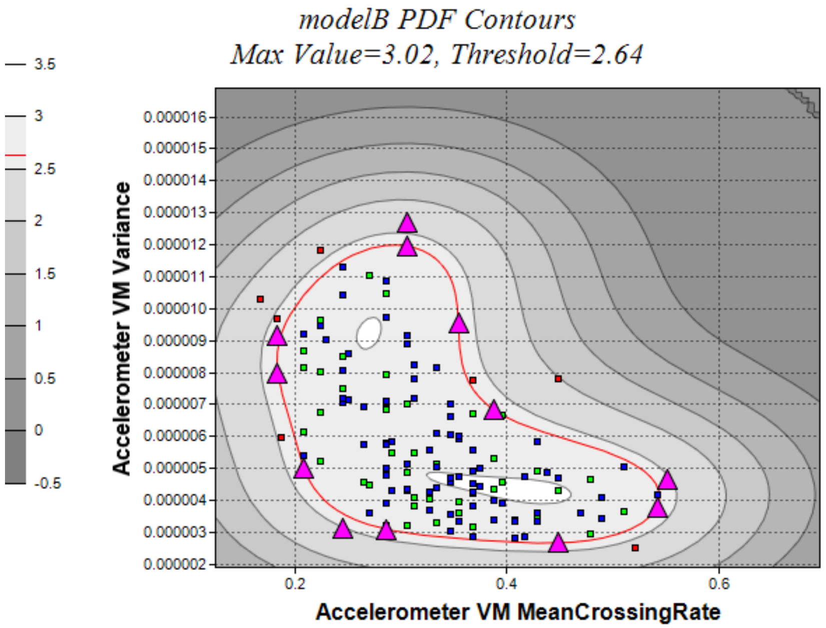
It is possible to train one-class models in situ on the MCU. Multiclass models are typically trained off-line

One Class Support Vector Machines

- Used for anomaly detection
- This algorithm tells us if a sample is considered to be part of a known population or not

We are using a Gaussian Kernel

$$\boldsymbol{f_{svm}(x) = \sum_{i=1}^{n} \alpha_i e^{-\frac{\|x - sv_i\|^2}{2\sigma^2}}}$$



where:

x is a d-dimensional feature vector

$\{sv_i\}_{i=1}^{n}$ are support vectors (SVs)

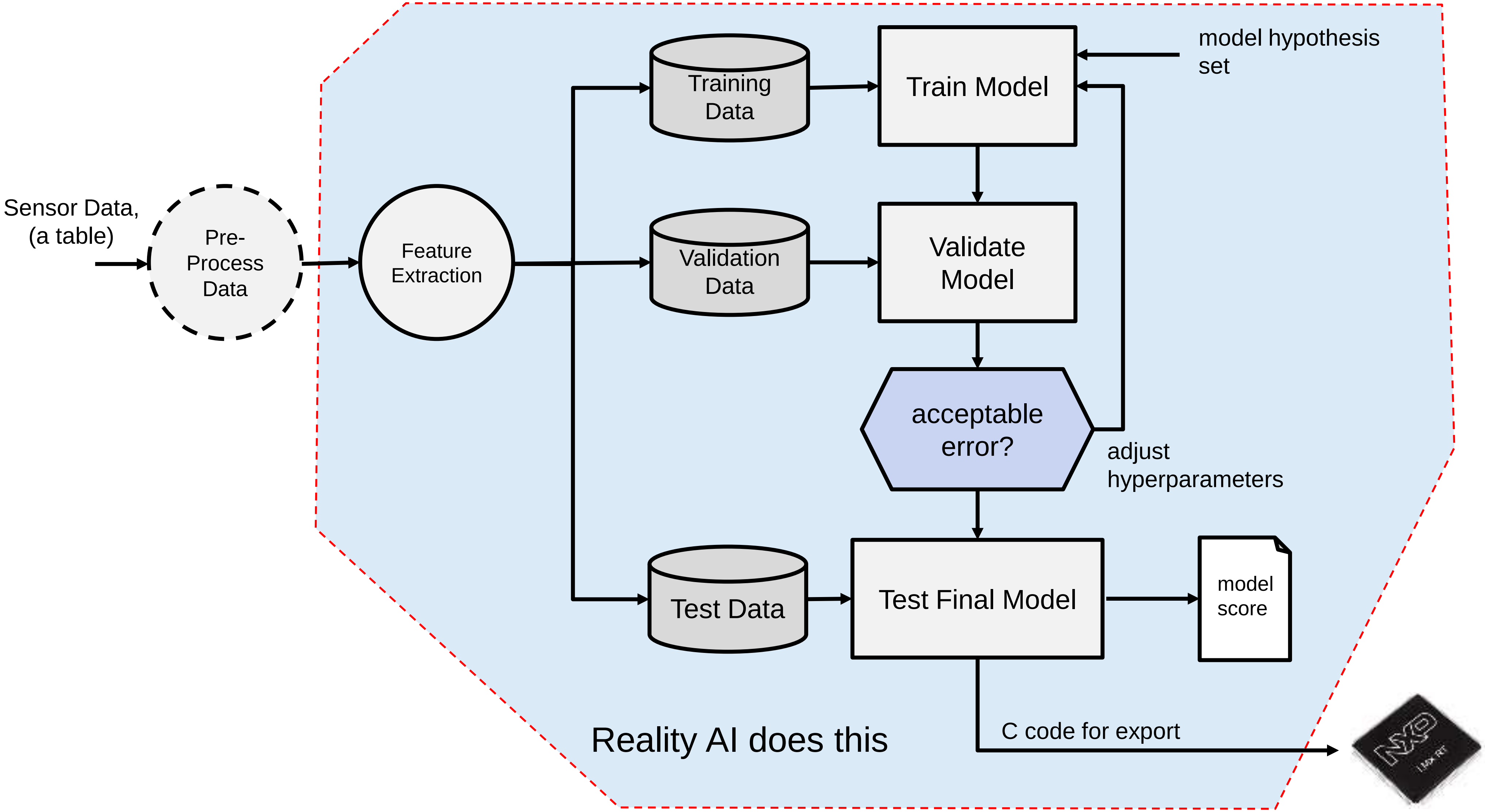
$\{\alpha_i\}_{i=1}^{n}$ are coefficients for SVs

σ is known as the “kernel size”

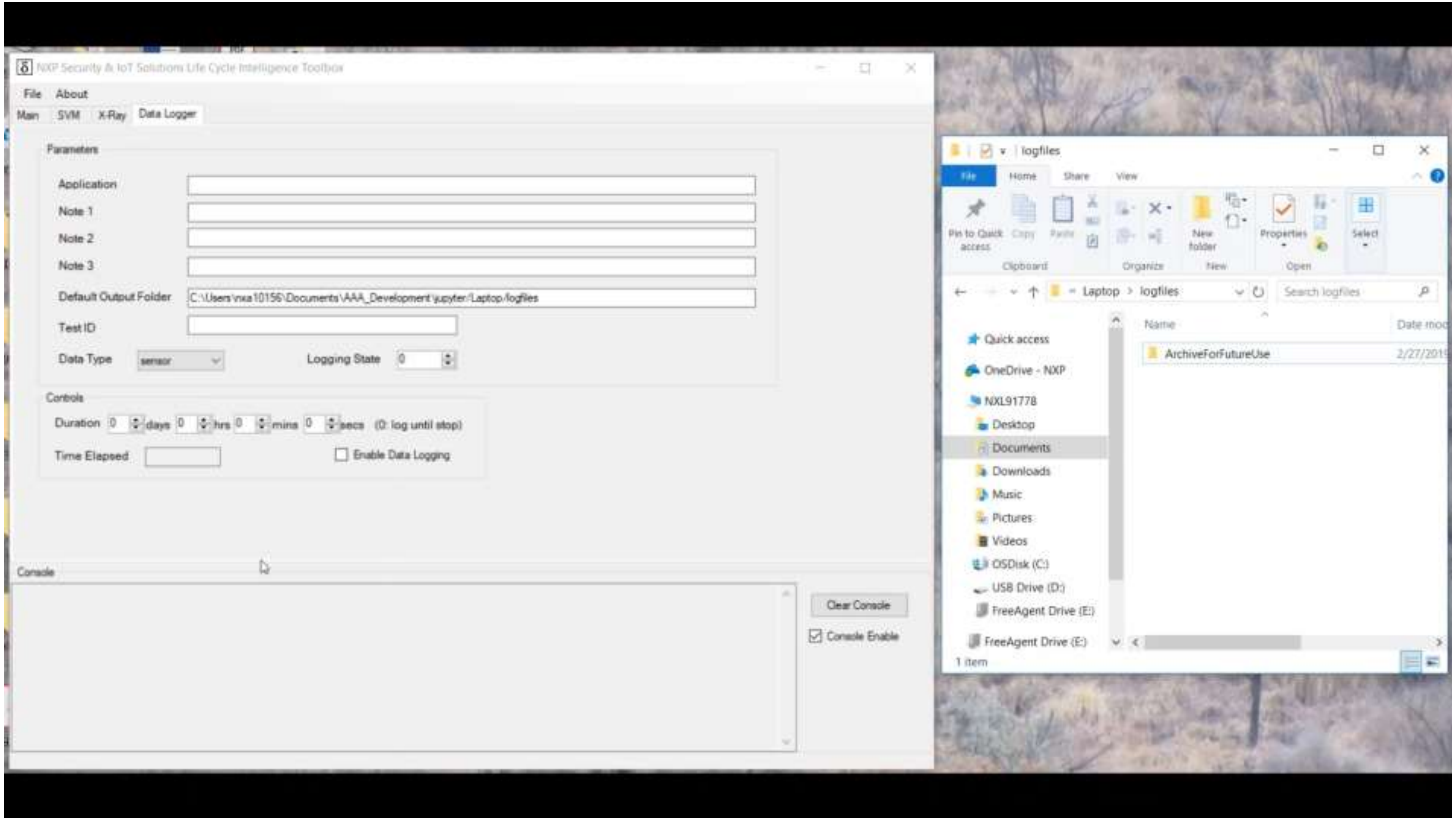
a sample is considered True (1) if pdf $f_{svm}(x)$ > threshold and False (-1) otherwise

$$sgn[f_{svm}(x) - thr]$$

Classical ML Workflow



A Sample Data Logging Session



NXP Security & IoT Solutions Life Cycle Intelligence Toolbox

Parameters

Application
Note 1
Note 2
Note 3
Default Output Folder: C:\Users\nxa10156\Documents\AAA_Development\jupyter\Laptop\logfiles
Test ID
Data Type: sensor
Logging State: 0

Controls

Duration 0 days 0 hrs 0 mins 0 secs (0: log until stop)
Time Elapsed
☐ Enable Data Logging

Console

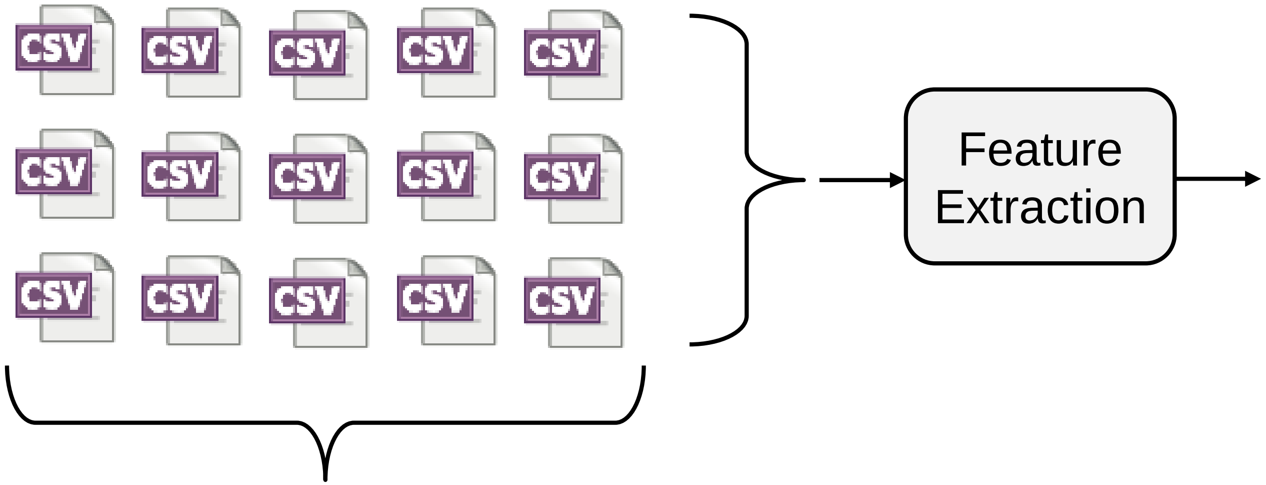
Clear Console
☑ Console Enable

logfiles

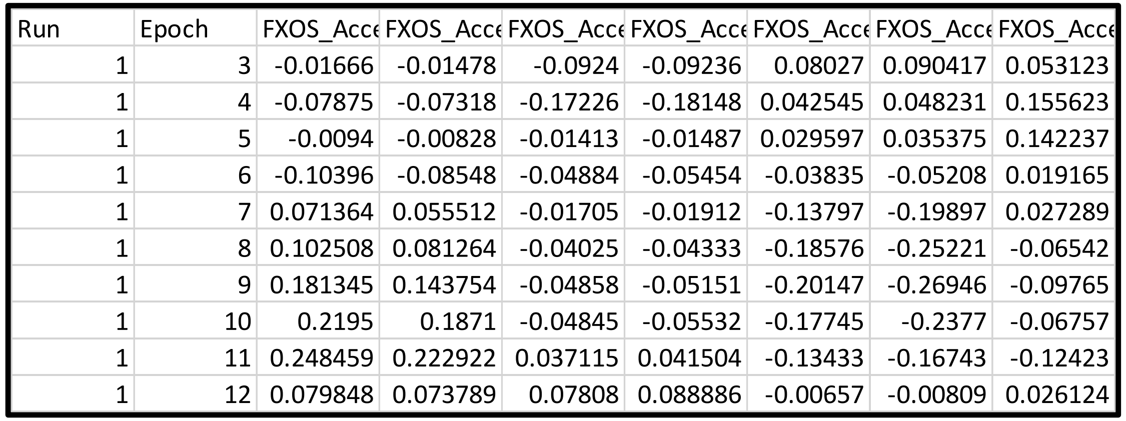
Name	Date mod
ArchiveForFutureUse	2/27/201

1 item

You can think of the resulting set of data as a (#runs X #sensors) array of .csv files.



data.csv



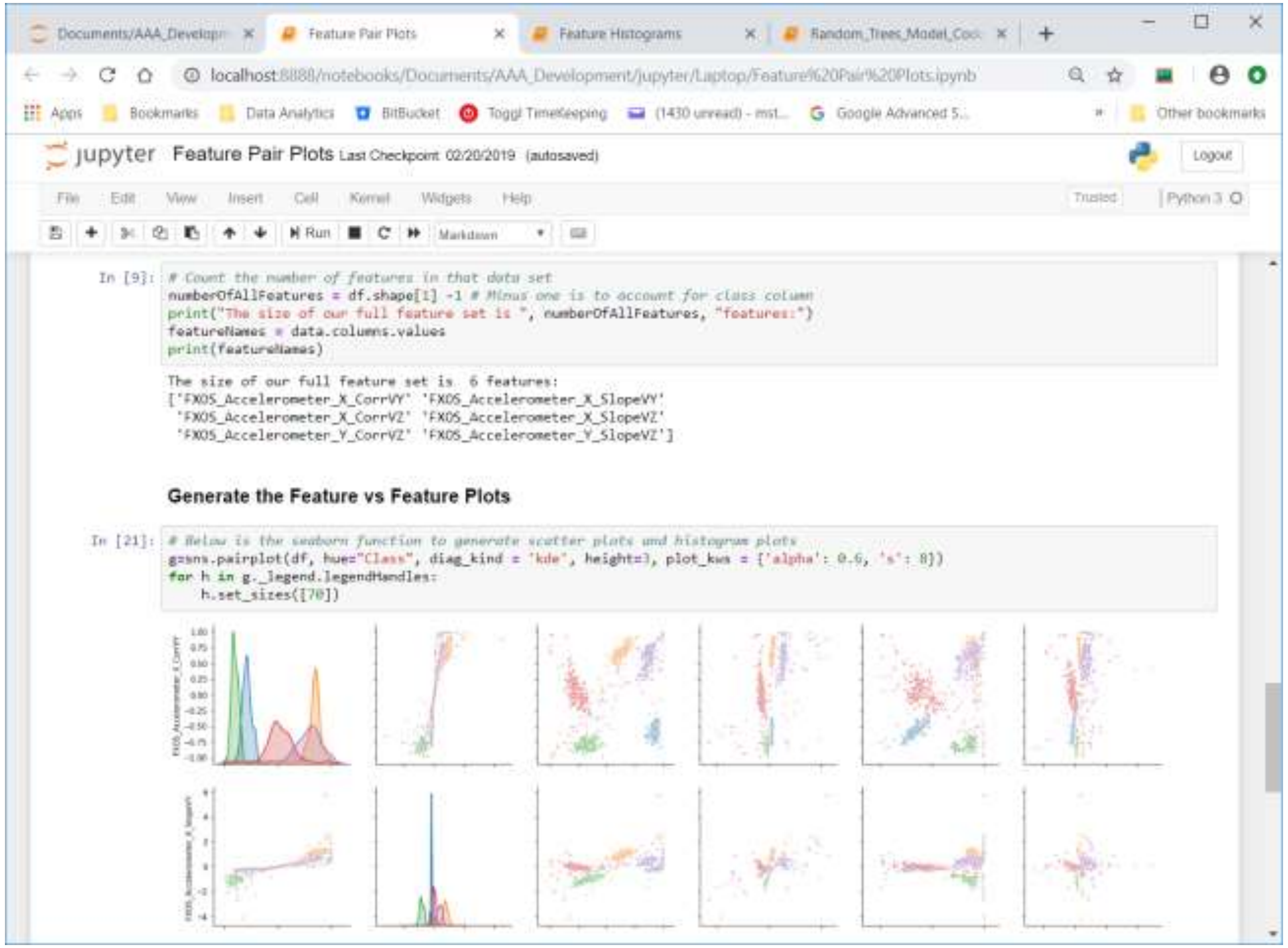
Run	Epoch	FXOS_Acce	FXOS_Acce	FXOS_Acce	FXOS_Acce	FXOS_Acce	FXOS_Acce	FXOS_Acce
1	3	-0.01666	-0.01478	-0.0924	-0.09236	0.08027	0.090417	0.053123
1	4	-0.07875	-0.07318	-0.17226	-0.18148	0.042545	0.048231	0.155623
1	5	-0.0094	-0.00828	-0.01413	-0.01487	0.029597	0.035375	0.142237
1	6	-0.10396	-0.08548	-0.04884	-0.05454	-0.03835	-0.05208	0.019165
1	7	0.071364	0.055512	-0.01705	-0.01912	-0.13797	-0.19897	0.027289
1	8	0.102508	0.081264	-0.04025	-0.04333	-0.18576	-0.25221	-0.06542
1	9	0.181345	0.143754	-0.04858	-0.05151	-0.20147	-0.26946	-0.09765
1	10	0.2195	0.1871	-0.04845	-0.05532	-0.17745	-0.2377	-0.06757
1	11	0.248459	0.222922	0.037115	0.041504	-0.13433	-0.16743	-0.12423
1	12	0.079848	0.073789	0.07808	0.088886	-0.00657	-0.00809	0.026124

Data Logger

- Features from all log files get consolidated into one “data.csv” file
- Notice that we are using a rolling window of 4 Epochs of raw data to compute 1 Epoch of features (hence we start at Epoch 3 above).
- This optimizes the number of feature samples obtained in each logging run.

Jupyter Notebooks

- Jupyter gives us the ability to execute R and Python scripts in a web browser environment
- Documentation, code and results are all part of a notebook
- Notebooks can easily be shared via email
- Python supports OpenCV, which means we can train models off-line and then export to our embedded environment



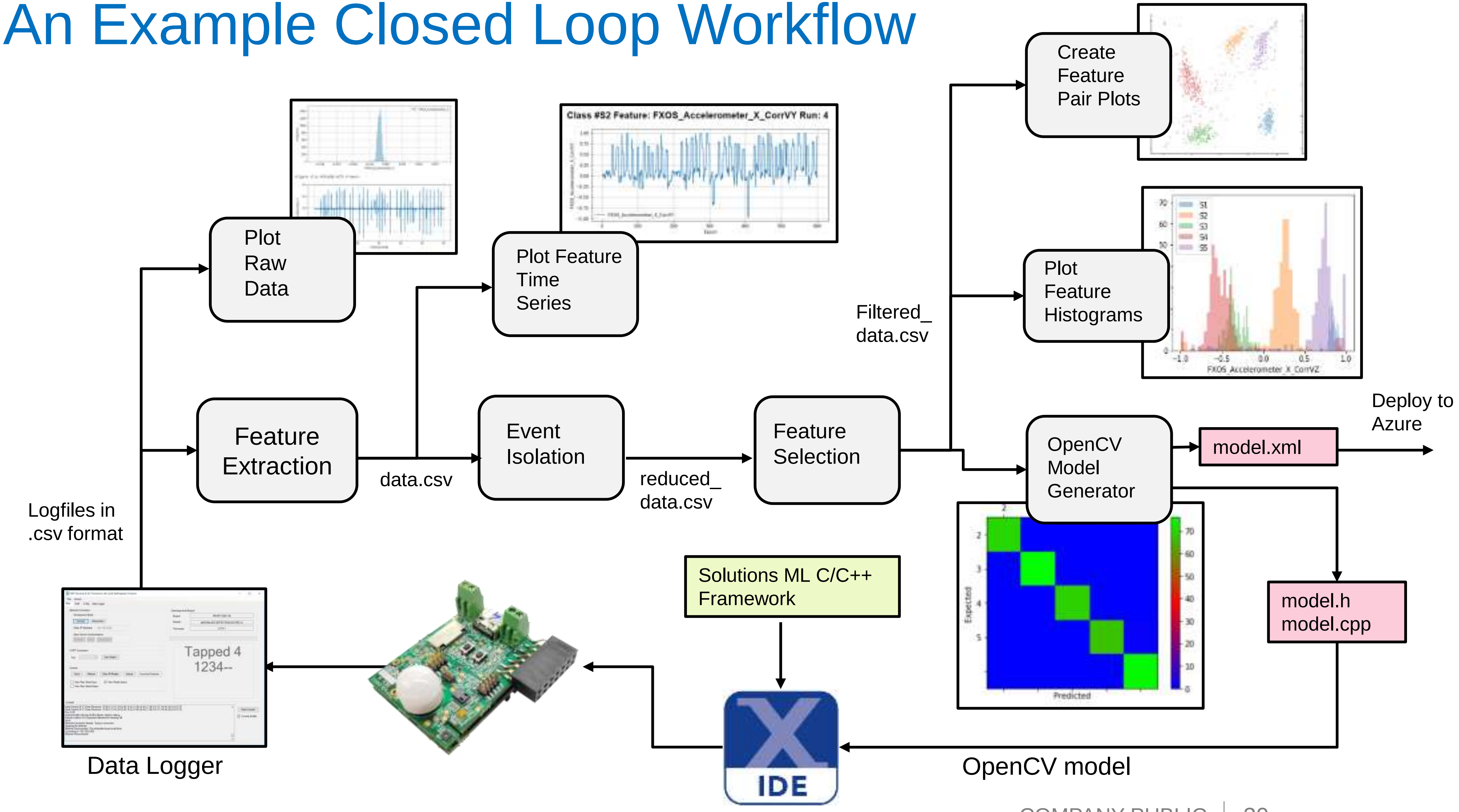
```
In [9]: # Count the number of features in that data set
        numberOfAllFeatures = df.shape[1] -1 # Minus one is to account for class column
        print("The size of our full feature set is ", numberOfAllFeatures, "features:")
        featureNames = data.columns.values
        print(featureNames)

        The size of our full feature set is  6 features:
        ['FXOS_Accelerometer_X_CorrVY' 'FXOS_Accelerometer_X_SlopeVY'
         'FXOS_Accelerometer_X_CorrVZ' 'FXOS_Accelerometer_X_SlopeVZ'
         'FXOS_Accelerometer_Y_CorrVZ' 'FXOS_Accelerometer_Y_SlopeVZ']
```

Generate the Feature vs Feature Plots

```
In [21]: # Below is the seaborn function to generate scatter plots and histogram plots
         g=sns.pairplot(df, hue="Class", diag_kind = 'kde', height=3, plot_kws = {'alpha': 0.6, 's': 8})
         for h in g._legend.legendHandles:
             h.set_sizes([70])
```

An Example Closed Loop Workflow



Plot Raw Data

Plot Feature Time Series

Create Feature Pair Plots

Plot Feature Histograms

Feature Extraction

data.csv

Event Isolation

reduced_ data.csv

Feature Selection

Filtered_ data.csv

OpenCV Model Generator

model.xml

Deploy to Azure

model.h model.cpp

Logfiles in .csv format

Solutions ML C/C++ Framework

Data Logger

OpenCV model

Class #S2 Feature: FXOS_Accelerometer_X_CorrVY Run: 4
S1
S2
S3
S4
S5
FXOS_Accelerometer_X_CorrVZ
Tapped 4
1234--
Expected
Predicted
IDE

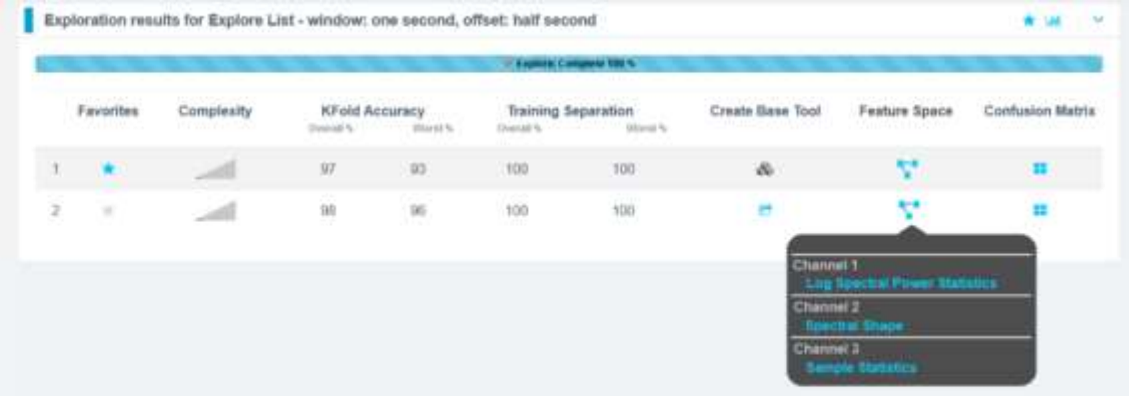
Reality AI Option

Reality AI®

Explainable AI at the Edge

Reality AI Offers:

- Data Collection & Labelling Services
- Reality AI Tools® Software
- Ready to Customize Solutions



Exploration results for Explore List - window: one second, offset: half second

Explore Complete 100 %

	Favorites	Complexity	KFold Accuracy		Training Separation		Create Base Tool	Feature Space	Confusion Matrix
			Overall %	Worst %	Overall %	Worst %			
1	★		97	93	100	100			
2	☆		98	96	100	100			

Channel 1
Log Spectral Power Statistics
Channel 2
Spectral Shape
Channel 3
Sample Statistics

Reality AI models sit on top of the CMLT sensor drivers and are fully compatible with the CMLT hardware and software.

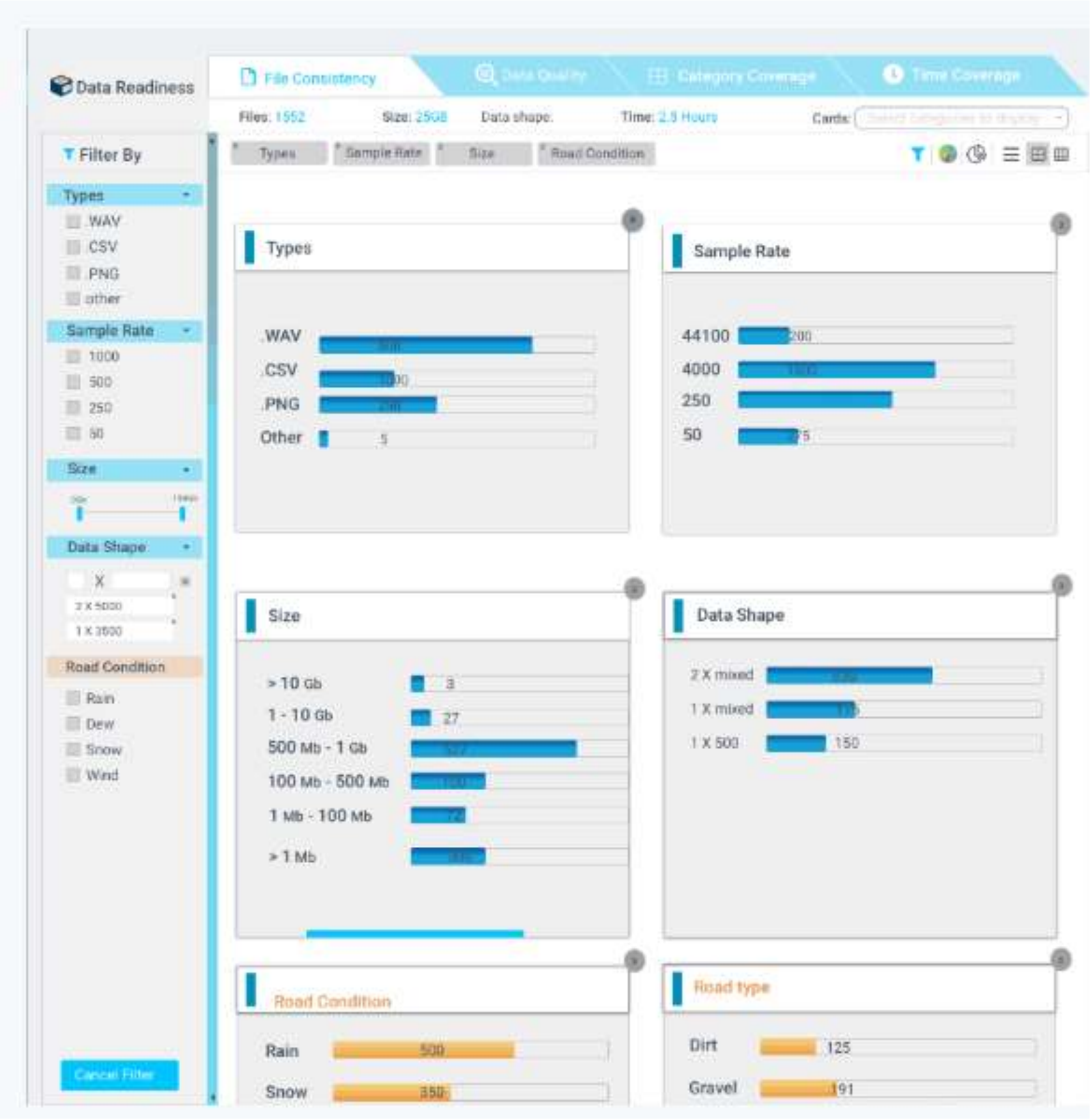
Reality AI provides a optimized path to production for those companies who don’t have the desire or time to come up the ML learning curve.

Reality AI®

Explainable AI at the Edge

Reality AI Data Collection Services:

- Data collection planning
- Data collection hardware design
- Proof-of-Concept planning and structuring
- Data labeling
- Data consistency and quality assessment
- Data analysis and model construction using Reality AI Tools®
- Performance testing and tuning
- Explainable AI: time-frequency analysis, factor analysis and decision significance testing using Reality AI Tools

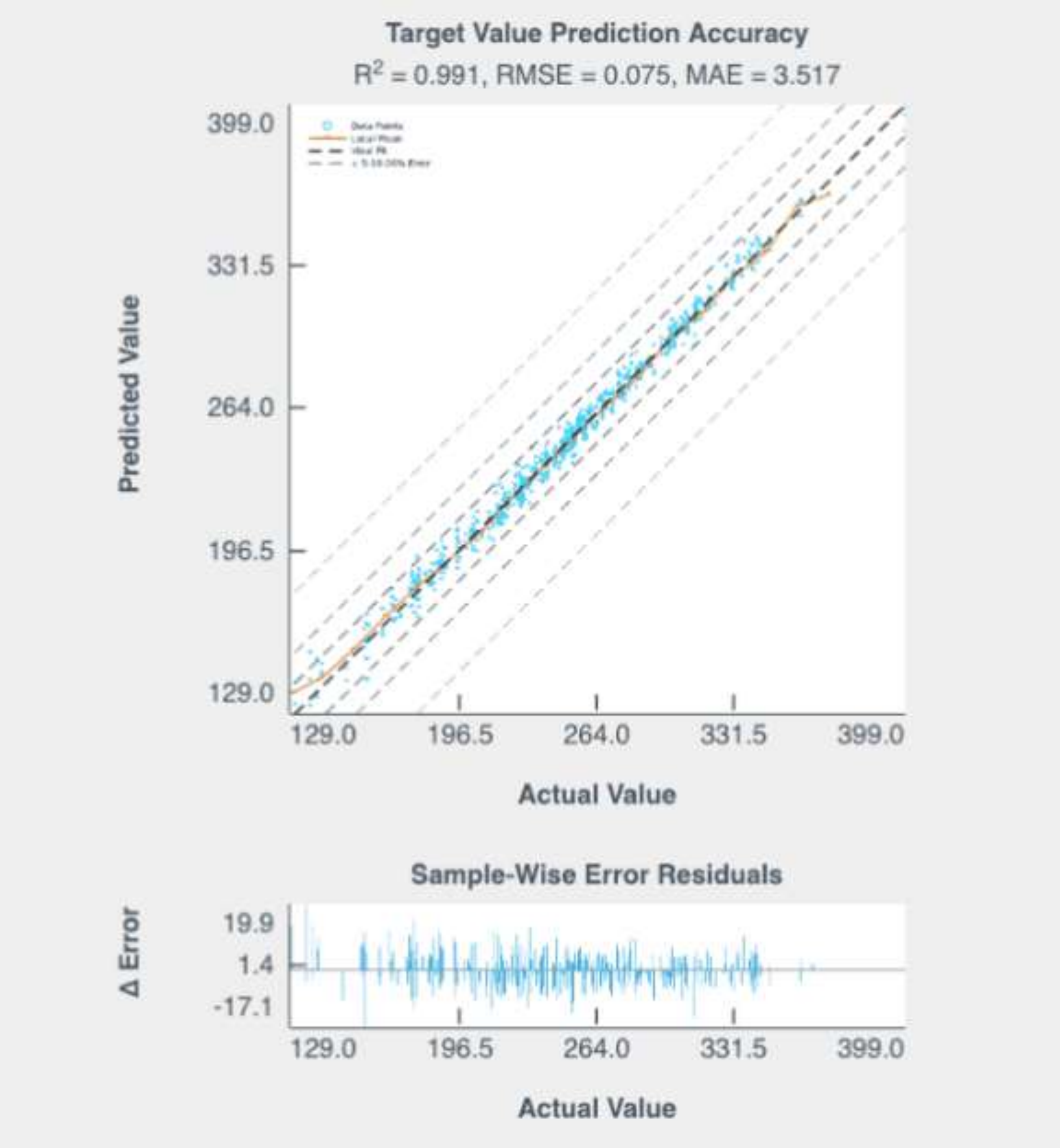

Files: 1552 Size: 25GB Data shape: Time: 2.5 Hours Cards:

Reality AI®

Explainable AI at the Edge

Reality AI Tools®:

- Discovers optimized features from your data.
- Automatically generates machine learning models using discovered features
- Allows for cloud-based testing and validation, before exporting to embedded environments
- Generates inference code, compiled for your specific target, allowing machine learning models to run on edge devices

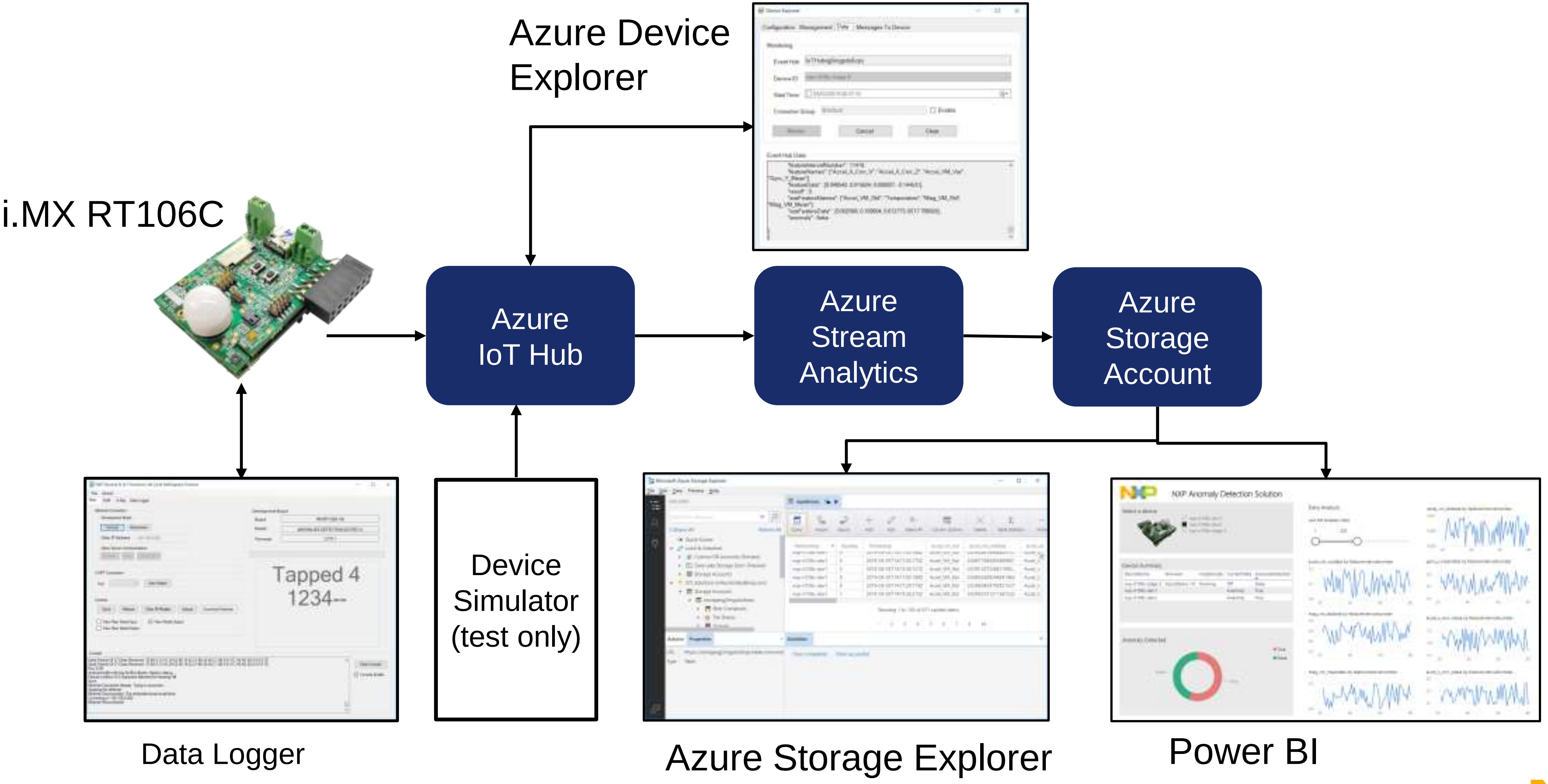


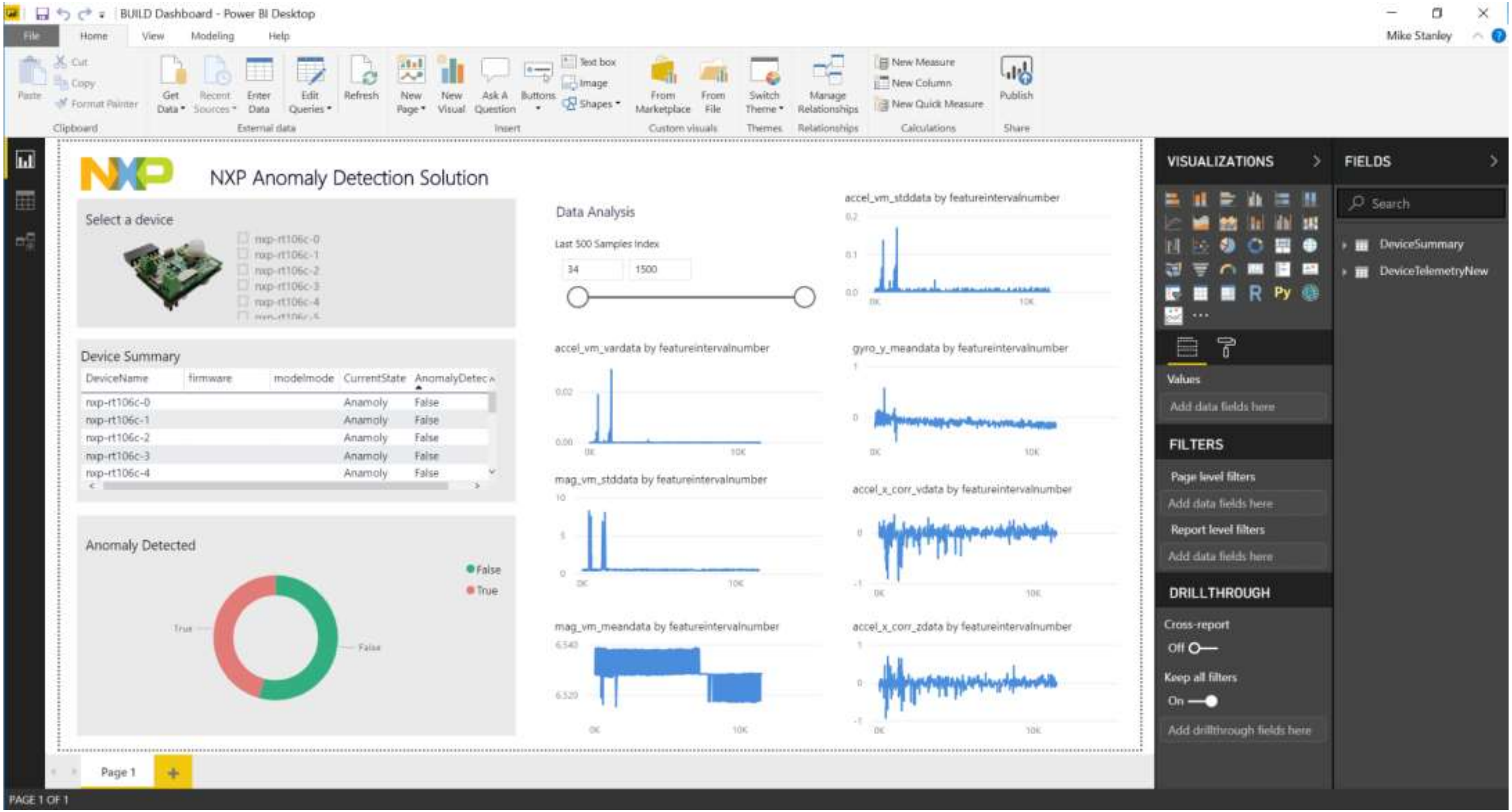
Microsoft Azure Interfaces (Under Development)

We Can Extend our ML Solution into the Cloud Using Azure

- Azure Machine Learning supports Python and the same OpenCV library functions we are using on our MCU
- Models can be pushed or pulled for Azure
- Models can be retrained on Azure
- Azure lets us scale our solution to large populations of fielded products, while retaining the ability to monitor and/or perform OTA updates.

Azure Data Stream





Device Summary

DeviceName	firmware	modelmode	CurrentState	AnomalyDetec
nxp-rt106c-0			Anamoly	False
nxp-rt106c-1			Anamoly	False
nxp-rt106c-2			Anamoly	False
nxp-rt106c-3			Anamoly	False
nxp-rt106c-4			Anamoly	False

The plan is to replace this Power.BI dashboard with an IoT Central Implementation

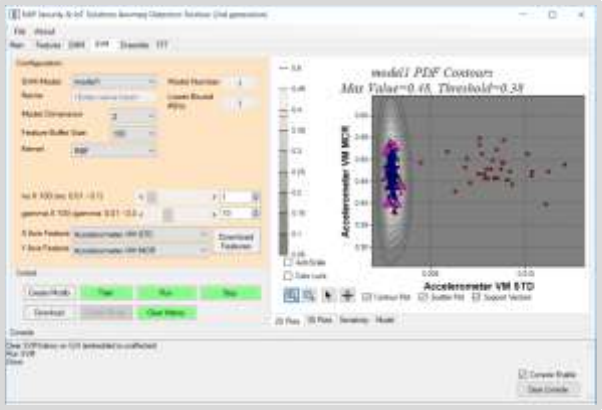
Microsoft Integration Status

- Partnership announced at BUILD 2019
- Proof of Concept also demonstrated at BUILD 2019
- Plug and Play and Device Provisioning features must be added before it is viable to hand this off to customers. Plan for that to be developed in 3Q

Availability & Contacts

Solution Recap

Off the shelf hardware for sensor data collection and product prototyping.



Windows-based tool for data collection for machine learning and measuring application metrics

Expert 3rd party SaaS models can be directly integrated into the framework.

Reality AI

AI at the Edge

Embedded firmware engine for machine learning delivered as a custom SDK

SDK

Coming soon…

Microsoft Azure

Take advantage of Azure cloud infrastructure and ML

jupyter

OpenCV

python

Easily integrate models trained via standard open source tools.

Pre-release Hardware and Software Available Under NDA

Please contact Bill.Krakar@nxp.com to initiate access. Ask for information about the i.MX RT106C and associated toolkit.

Technical Leads:

Michael Stanley

- 38 years experience in the semiconductor field at Motorola, Freescale and NXP in areas ranging from circuit design to machine learning
- 8 patents and numerous publications, blog postings, etc.
- Contributor to Measurement, Instrumentation and Sensors Handbook, 2nd edition
- Senior Member of the IEEE and IEEE Standard 2700-2014 contributor

Dr. Jongmin Lee

- Systems & Architecture Engineer at NXP
- Interests are statistical signal processing and machine learning

Mike and Jongmin are the authors of “*Sensor Analysis for the Internet of Things*”, published last year by Morgan & Claypool Publishers.

SECURE CONNECTIONS
FOR A SMARTER WORLD