



**Internet of Things (IoT) in Cities, Industry, Business
– A Case Study Approach**

FIOT-1-C2899

Project Report

Submitted by

LIM YUAN HER (17060167)

Table of Contents

| | | |
|---------|--|----|
| 1. | Introduction..... | 1 |
| 2. | Design Prototyping | 1 |
| 2.1 | Objective..... | 1 |
| 2.2 | Background..... | 1 |
| 2.3 | System Architecture | 2 |
| 2.4 | Data Flow | 2 |
| 2.4.1 | IoT Data Metrics..... | 2 |
| 2.4.2 | Information Flow | 2 |
| 2.5 | Experimental Setup..... | 3 |
| 2.5.1 | Test Setup | 3 |
| 2.5.1.1 | Lap Run 1 | 4 |
| 2.5.1.2 | Lap Run 2 | 4 |
| 2.5.1.3 | Lap Run 3 | 5 |
| 2.5.1.4 | Lap Run Results | 6 |
| 3. | Data Visualization | 6 |
| 3.1 | ThingWorX | 6 |
| 4. | Data Analytics | 8 |
| 4.1 | Problem Definition..... | 8 |
| 4.1.1 | Introduction | 8 |
| 4.1.2 | Data Summarization | 8 |
| 4.1.2.1 | Define Dataset | 8 |
| 4.1.2.2 | Descriptive Statistics | 8 |
| 4.1.3 | Data Preparation..... | 10 |
| 4.1.3.1 | Feature Selection | 10 |
| 4.1.4 | Clustering Evaluation | 10 |
| 4.1.4.1 | Select Optimal K..... | 10 |
| 4.1.4.2 | Evaluate Algorithm Performance..... | 11 |
| 4.1.4.3 | Evaluate Algorithm Performance on Validation/ Test Dataset | 13 |
| 4.1.4.4 | Lap Timing Derivation | 14 |
| 4.1.4.5 | Overall Equipment Effectiveness (OEE) Calculation | 14 |
| 4.1.5 | Time Series Analysis..... | 15 |
| 4.1.5.1 | Introduction | 15 |
| 4.1.5.2 | SARIMA Analysis | 15 |

| | | |
|---------|---|----|
| 4.1.6 | Multi-Class Classification | 17 |
| 4.1.6.1 | Introduction | 17 |
| 4.1.6.2 | Training/Validation Dataset and Evaluation Metric | 17 |
| 4.1.6.3 | Spot Check Algorithms | 17 |
| 4.1.6.4 | Compare Algorithms | 17 |
| 4.1.6.5 | Estimate accuracy on validation dataset..... | 18 |
| 4.1.6.6 | Algorithm Tuning | 18 |
| 4.1.6.7 | Ensemble Methods | 18 |
| 4.1.6.8 | Finalize Model | 21 |
| 4.1.6.9 | Summary | 21 |
| 5. | Reflections | 22 |
| 5.1 | Challenges/ Recommendations | 22 |
| 6. | Conclusion | 22 |
| 7. | References | 22 |
| 8. | Appendix 1 – Source Code Listing | 23 |
| 8.1 | XDK Workbench | 23 |
| 8.2 | Jupyter Notebook | 33 |

Table of Figures

| | |
|--|----|
| FIGURE 1 – SYSTEM ARCHITECTURE | 2 |
| FIGURE 2 – DATA FLOW DIAGRAM | 2 |
| FIGURE 3 – THOUZER ROBOT TEST RUN CIRCUIT | 3 |
| FIGURE 4 – THINGWORX MASHUP..... | 6 |
| FIGURE 5 – STREAMING DATA VIEW | 7 |
| FIGURE 6 – HISTORICAL DATA VIEW..... | 7 |
| FIGURE 7 – DATE/PARAMETER FILTER | 7 |
| FIGURE 8 – EXPORT FUNCTION | 7 |
| FIGURE 9 – DATA CLASSIFICATION PROCESS | 8 |
| FIGURE 10 – DATASET STRUCTURE | 9 |
| FIGURE 11 – HISTOGRAM/ DENSITY/ BOX PLOTS | 9 |
| FIGURE 12 – NUMBER OF CLUSTERS K SELECTION USING ELBOW METHOD | 11 |
| FIGURE 13 – LAP RUN 1 DATA CLASSIFICATION ALGORITHM EVALUATION FROM SCATTER PLOTS..... | 11 |
| FIGURE 14 – LAP RUN 1 DATA CLASSIFICATION ALGORITHM EVALUATION FROM TIME-SERIES PLOTS..... | 12 |
| FIGURE 15 – SEGMENT IDENTIFICATION VIA VISUAL SCANNING | 12 |
| FIGURE 16 – CLUSTERING PERFORMANCE ON VALIDATION DATASET (LAP RUN 2) | 13 |
| FIGURE 17 – CLUSTERING PERFORMANCE ON TEST DATASET (LAP RUN 3) | 13 |
| FIGURE 18 –SERIES AND SEASONAL DECOMPOSITION PLOT OF LAP RUN 1 SEGMENT TIMINGS..... | 16 |
| FIGURE 19 –ACF/ PACF PLOTS FOR LAP RUN 1 SEGMENT TIMINGS | 16 |
| FIGURE 20 –FORECASTED/ LAP RUN 2 SEGMENT TIMING OVERLAY PLOT | 16 |
| FIGURE 21 –ALGORITHM SPOT CHECK RESULTS | 17 |
| FIGURE 22 –ALGORITHM COMPARISON ON TRANSFORMED DATA RESULTS | 17 |
| FIGURE 23 –VALIDATION DATASET ACCURACY ESTIMATION RESULTS | 18 |
| FIGURE 24 –ALGORITHM TUNING ON DECISION TREE CLASSIFIER RESULTS | 18 |
| FIGURE 25 –ENSEMBLE METHODS ALGORITHM SPOT CHECK RESULTS..... | 19 |
| FIGURE 26 – ENSEMBLE METHODS ALGORITHM COMPARISON ON TRANSFORMED DATA RESULTS..... | 19 |
| FIGURE 27 –ENSEMBLE METHOD VALIDATION DATASET ACCURACY ESTIMATION RESULTS | 20 |
| FIGURE 28 –ALGORITHM TUNING ON ADABOOST CLASSIFIER RESULTS | 20 |
| FIGURE 29 –ALGORITHM TUNING ON GRADIENT BOOSTING CLASSIFIER RESULTS | 20 |
| FIGURE 30 –PREDICTION RESULT ON TEST DATASET | 21 |

List of Tables

| | |
|--|----|
| TABLE 1 – IOT DATA METRICS | 2 |
| TABLE 2 –LAP RUN #2 | 4 |
| TABLE 3 –LAP RUN #3 | 5 |
| TABLE 4 –LAP RUN #1 | 6 |
| TABLE 5 –LAP RUN RESULTS STORAGE | 6 |
| TABLE 6 – DATA CLASSIFICATION TRAINING/ VALIDATION DATASETS..... | 8 |
| TABLE 7 – FEATURE SELECTION | 10 |
| TABLE 8 – BMI160 MEASUREMENTS FEATURE CONSIDERATION | 10 |
| TABLE 9 – LAP TIMINGS FOR LAP RUN 1/2/3 | 14 |
| TABLE 10 – SUPPORTING VARIABLES USED FOR OEE CALCULATION | 15 |
| TABLE 11 – OEE CALCULATION | 15 |

1. Introduction

This report details the design and development of an IoT solution for a Target Guided Autonomous Robotic Cart (Thouzer) produced by Doog Inc. company using a Bosch XDK110 universal programmable sensor device for telemetry monitoring and using the ThingWorX cloud IoT platform for data visualization.

In Section 2, we detail the design prototyping process and experimental setup.

In Section 3, we detail the use of the ThingWorX IoT Platform to perform data visualization of the sensor data.

Finally, in Section 4, we use machine learning techniques to perform predictive analytics on the sensor data for

2. Design Prototyping

2.1 Objective

The objective is to build an IoT solution using a Bosch XDK110 universal sensor device for gathering data from the Thouzer Autonomous Robotic Cart and visualizing the sensor using the ThingWorX IoT platform.

The purpose is to provide a mechanism for monitoring the robot telemetry and performance metrics. These include:

- Display real-time values of robot telemetry.
- Notification about significant performance changes or events.

2.2 Background

Infinity Assembly has recently invested in a collaborative robot (Thouzer Target Guided Autonomous Robotic Cart) which is used to transport product(s) by traversing a working area from a starting point to a destination by following a pre-planned route lined with reflective tape. The potential issue that can occur is that obstacles can occur in the path of travel that could slow down or stop the robot as it is programmed for collision avoidance. This could negatively affect the performance and efficiency of the robot.

The sensor data collected from the XDK110 sensor device attached to the autonomous robotic cart could be analysed to provide alerts of such degradation of performance in advance so that the necessary steps can be taken to investigate the cause earlier. This will help to improve the productivity of the robotic cart with reduced downtime.

To validate the proposed solution, experiments will be conducted with the collected data analysed using machine learning techniques to provide insights which will assist to improve the Overall Equipment Effectiveness (OEE) index.

2.3 System Architecture

Figure 1 below shows the overall system architecture:

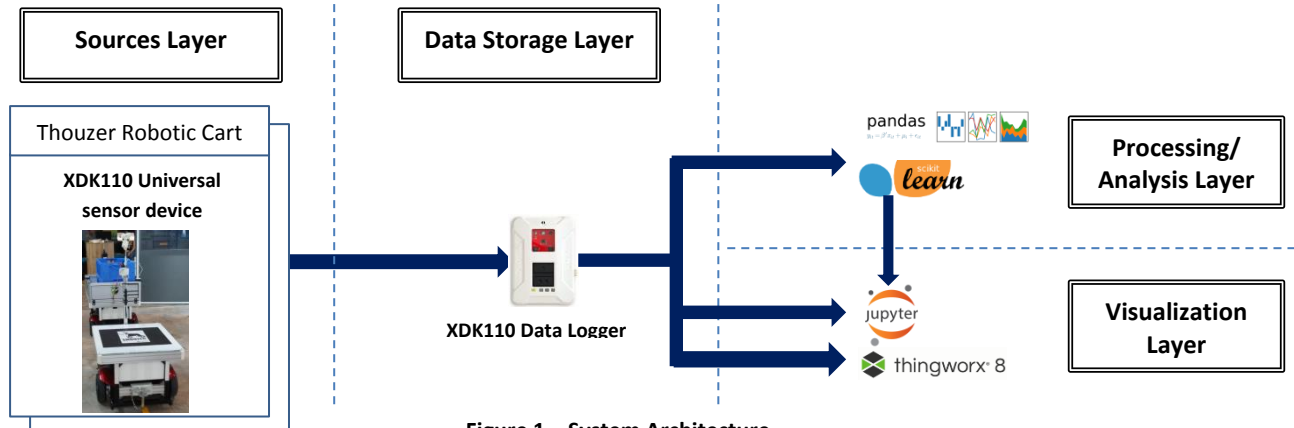


Figure 1 – System Architecture

The solution involves attaching a XDK110 sensor device to the autonomous robotic cart for telemetry data collection and transmitting the data to ThingWorX IoT platform wirelessly for real-time data visualization and monitoring and performing machine learning using Scikit-Learn to perform clustering and multi-class classification on the collected data.

2.4 Data Flow

2.4.1 IoT Data Metrics

The table below illustrates the data metrics collected from the XDK110 Universal Sensor Device:

| Sensor | Data | Description |
|--|--|---|
| 3-Axis Accelerometer BMA280 sensor | bma280_x, bma280_y, bma280_z | Linear Acceleration (X-, Y-, Z-axis) |
| 3-Axis Gyroscope BMG160 sensor | bmg160_x, bmg160_y, bmg160_z | Angular Velocity (X-, Y-, Z-axis) |
| Magnetometer BMM150 sensor | bmm150_x, bmm150_y, bmm150_z | Magnetic Field (X-, Y-, Z-axis) |
| Environmental (Temperature/ Humidity/ Pressure) BME280 sensor | bme280_temp, bme280_hum, bme280_press | Temperature Relative Humidity Atmospheric Pressure |
| Inertial Measurement Unit BMI160 (3-Axis Gyroscope (Open Loop) + 3-Axis Accelerometer) | bmi160_g_x, bmi160_g_y, bmi160_g_z bmi160_a_x, bmi160_a_y, bmi160_a_z | Angular Velocity (X-, Y-, Z-axis) Linear Acceleration (X-, Y-, Z-axis) |

Table 1 – IoT Data Metrics

2.4.2 Information Flow

The figure below illustrates the flow of data from the XDK110:

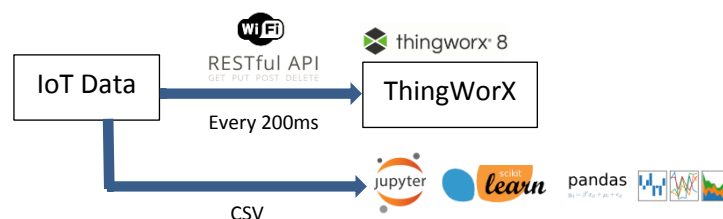


Figure 2 – Data Flow Diagram

The data is periodically sent (every 2s) using POST API messages sent wirelessly from the XDK110 sensor device to ThingWorX. Alternatively, sensor data can be stored in the micro-SD card in the XDK110 device in comma-separated variable (CSV) format using the data logger function. Machine learning techniques can also be performed on the collected data for analysis purposes to correlate the physical state of the autonomous robotic cart e.g. veer left/ right, start/stop etc.

Data is visualized using a Mashup display create in the ThingWorX IoT platform.

2.5 Experimental Setup

The first step in this process is to collect data from the autonomous robotic cart using the XDK sensor device, and this involves conducting lap runs of the autonomous robotic cart along a pre-planned route lined with reflective tape. The lap runs are designed such that a standard baseline can be established for performance evaluation and to simulate abnormal conditions as follows:

| Lap Run | Description | Remark |
|---------|---|---|
| 1 | Normal run in counter-clockwise direction | Establish Baseline |
| 2 | Abnormal run in counter-clockwise direction with obstacle | Simulate abnormal condition of frequent stoppages |
| 3 | Normal run in clockwise direction | - |

Based on the data collected from test runs (1), the performance baseline is established from which relevant machine learning technique(s) can be employed to identify performance deviation using the data collected in test run (2).

The data collection from test run (3) is used to further validate the developed model.

2.5.1 Test Setup

Figure 3 below shows the Thouzer Robotic Cart test run circuit and mounting location of the XDK110 universal sensor device:



Figure 3 – Thouzer Robot Test Run Circuit

The circular lap run circuit consists of 10 left/ right turn segments (see illustration above) and the Thouzer Autonomous Robotic Cart is made to do 10 laps for each lap run.

2.5.1.1 Lap Run 1

Lap Run 1 consists of 9 counter-clockwise laps through an even travel path with no obstacles in between.



























| Segment | From | | To | |
|---------|---|---|--|---|
| 1 |  | 1 st Turn Left (Start of Lap) |  | 2 nd Turn Left |
| 2 |  | 2 nd Turn Left |  | 3 rd Turn Left |
| 3 |  | 3 rd Turn Left |  | 4 th Turn Left |
| 4 |  | 4 th Turn Left |  | 5 th Turn Left |
| 5 |  | 5 th Turn Left |  | 1 st Turn Right |
| 6 |  | 1 st Turn Right |  | 6 th Turn Left |
| 7 |  | 6 th Turn Left |  | 7 th Turn Left |
| 8 |  | 7 th Turn Left |  | 8 th Turn Left |
| 9 |  | 8 th Turn Left |  | 9 th Turn Left |
| 10 |  | 9 th Turn Left |  | 1 st Turn Left (End of Lap) |

Table 2 –Lap Run #2

2.5.1.2 Lap Run 2

Lap Run 3 consists of 9 counter-clockwise laps through an even travel path simulating frequent stoppages using a person to block the Thouzer Autonomous Robotic Cart.

| Segment | From | | To | |
|---------|---|---|--|---------------------------|
| 1 |  | 1 st Turn Left (Start of Lap) |  | 2 nd Turn Left |
| 2 |  | 2 nd Turn Left |  | 3 rd Turn Left |
| 3 |  | 3 rd Turn Left |  | 4 th Turn Left |














| Segment | From | | To | |
|---------|---|----------------------------|--|---|
| 4 |  | 4 th Turn Left |  | 5 th Turn Left |
| 5 |  | 5 th Turn Left |  | Barrier #1 |
| |  | Barrier #1 |  | 1 st Turn Right |
| 6 |  | 1 st Turn Right |  | 6 th Turn Left |
| 7 |  | 6 th Turn Left |  | 7 th Turn Left |
| 8 |  | 7 th Turn Left |  | 8 th Turn Left |
| 9 |  | 8 th Turn Left |  | Barrier #2 |
| 10 |  | Barrier #2 |  | 9 th Turn Left |
| |  | 9 th Turn Left |  | 1 st Turn Left (End of Lap) |

Table 3 –Lap Run #3

2.5.1.3 Lap Run 3

Lap Run 3 consists of 10 clockwise laps through an even travel path with no obstacles in between.

| Segment | From | | To | |
|---------|---|--|--|----------------------------|
| 1 |  | 1 st Turn Right (Start of Lap) |  | 2 nd Turn Right |
| 2 |  | 2 nd Turn Right |  | 3 rd Turn Right |
| 3 |  | 3 rd Turn Right |  | 4 th Turn Right |
| 4 |  | 4 th Turn Right |  | 1 st Turn Left |
| 5 |  | 1 st Turn Left |  | 5 th Turn Right |
| 6 |  | 5 th Turn Right |  | 6 th Turn Right |









| Segment | From | | To | |
|---------|---|----------------------------|--|--|
| 7 |  | 6 th Turn Right |  | 7 th Turn Right |
| 8 |  | 7 th Turn Right |  | 8 th Turn Right |
| 9 |  | 8 th Turn Right |  | 9 th Turn Right (End of Lap) |
| 10 |  | 9 th Turn Right |  | 1 st Turn Right (End of Lap) |

Table 4 –Lap Run #1

2.5.1.4 Lap Run Results

The data collected from the test runs as identified in the previous section are stored in 4 comma-separated delimited files as below:

| Test Run | Data File name |
|----------|----------------|
| 1 | LOG_0004.CSV |
| 2 | LOG_0005.CSV |
| 3 | LOG_0003.CSV |

Table 5 –Lap Run Results Storage

Data files for lap run 1 will be used for model development to establish a performance baseline for the evaluation metric used i.e. lap timing.

We withhold data from lap run 2 from analysis and model selection and this data is used to validate the model. In addition, result from lap run 2 will be used to analyse test conditions used to simulate abnormalities separately.

3. Data Visualization

3.1 ThingWorX

A mashup display created in ThingWorX is used to visualize sensor data from the XDK110:

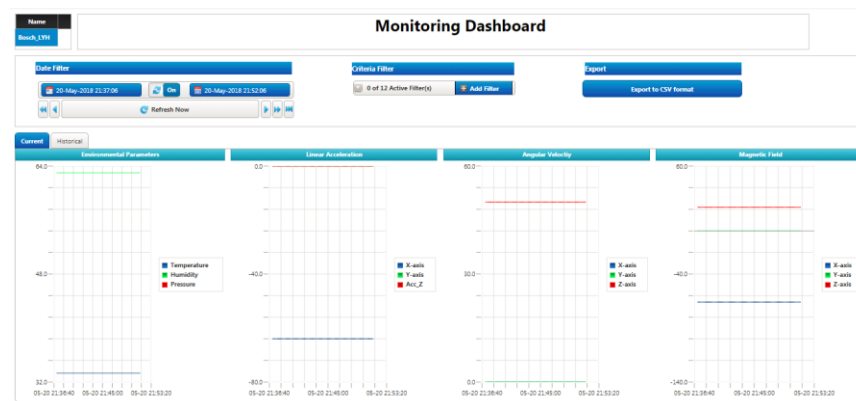


Figure 4 – ThingWorX Mashup

The dashboard provides the following functionality:

1. View streaming real-time data from the XDK110 sensor device.

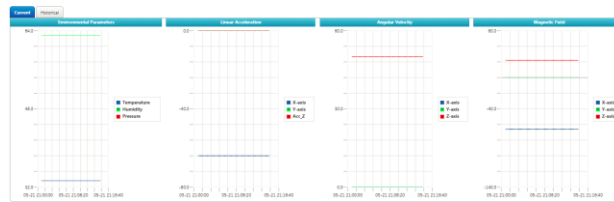


Figure 5 – Streaming Data View

2. View historical data for each sensor parameter.



Figure 6 – Historical Data View

3. Filter data by date range or sensor parameter value

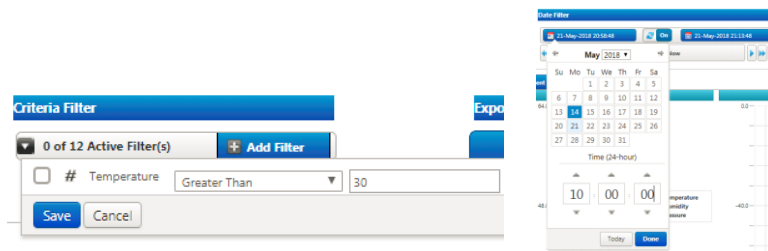


Figure 7 – Date/Parameter Filter

4. Export on-screen data to CSV format for offline analysis.

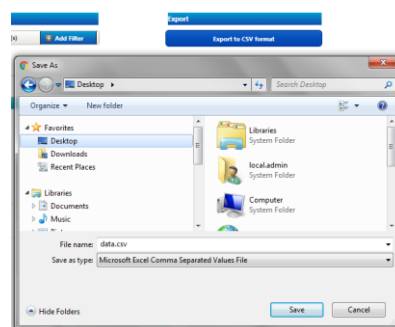


Figure 8 – Export Function

5. Listing of multiple XDK110 sensor devices attached to different autonomous robotic cart and on-screen data updating based on device selected for future expansion.

The benefit of using the ThingWorX IoT platform is that sensor data can be viewed from any desktop or mobile devices as long as internet connectivity is available.

4. Data Analytics

4.1 Problem Definition

4.1.1 Introduction

The problem is to develop a machine learning model which could be used to establish a performance baseline for the lap timings (using data from lap run (1) from which future performance deviation can be monitored (validated using data from lap runs (2) and (3)).

The sensor data recorded by the XDK110 device are unlabelled time-series data. Clustering, which is an unsupervised machine learning technique, is used to label the data. From the clustering results, we can derive the lap timings for each Lap Run. This is used to compute the operating time of the autonomous robotic cart that can be used in the calculation of the equipment's OEE. In addition, this performance deviation can be monitored against a rolling prediction made periodically to raise any alert notifications if the observed values fall within the low-confidence bands.

In addition, with the labelled data, multi-class classification, which is a supervised machine learning technique, is used to automatically classify the motion of the autonomous robotic cart for subsequent lap runs. This can be used for remote telemetry monitoring of the autonomous robotic cart as it moves through the factory floor transporting parts.

The figure below illustrates the process as described above:

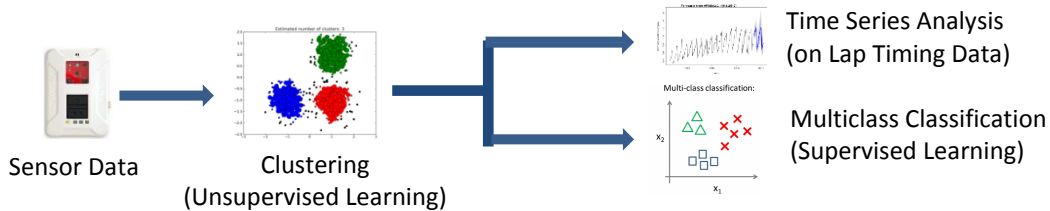


Figure 9 – Data Classification Process

4.1.2 Data Summarization

4.1.2.1 Define Dataset

The table below summarizes the source of the data used as the training and validation datasets for the subsequent machine learning model development/testing:

| Lap Run | Data File name | Dataset Type |
|---------|----------------|--------------|
| 1 | LOG_0004.CSV | Training |
| 2 | LOG_0005.CSV | Validation |

Table 6 – Data Classification Training/ Validation Datasets

Data from Lap Run 1 will be used for training and that from Lap Run 2 will be used for validation.

4.1.2.2 Descriptive Statistics

The dataset from lap run 1 consist of sensor data recorded by the XDK sensor device. Using the “dftypes” command from the pandas library provides a listing of the features of the dataset:

```
# types
print(df_LR1.dtypes)

unit--timestamp[ms]      int64
bma280_x[mg]             int64
bma280_y[mg]             int64
bma280_z[mg]             int64
bmg160_x[mDeg]           int64
bmg160_y[mDeg]           int64
bmg160_z[mDeg]           int64
bmi160_a_x[mg]           int64
bmi160_a_y[mg]           int64
bmi160_a_z[mg]           int64
bmi160_g_x[mDeg]        int64
bmi160_g_y[mDeg]        int64
bmi160_g_z[mDeg]        int64
bmm150_x[microT]         int64
bmm150_y[microT]         int64
bmm150_z[microT]         int64
bmm150_res               int64
bme280_temp[mDeg]        int64
bme280_press[Pa]         int64
bme280_hum[rh]           int64
max44009_bright[mLux]    int64
dtype: object
```

Dataset consists of 21 features in total (all integer type data)

Figure 10 – Dataset Structure

Below are histogram/ density/ box plots of the features in the dataset:

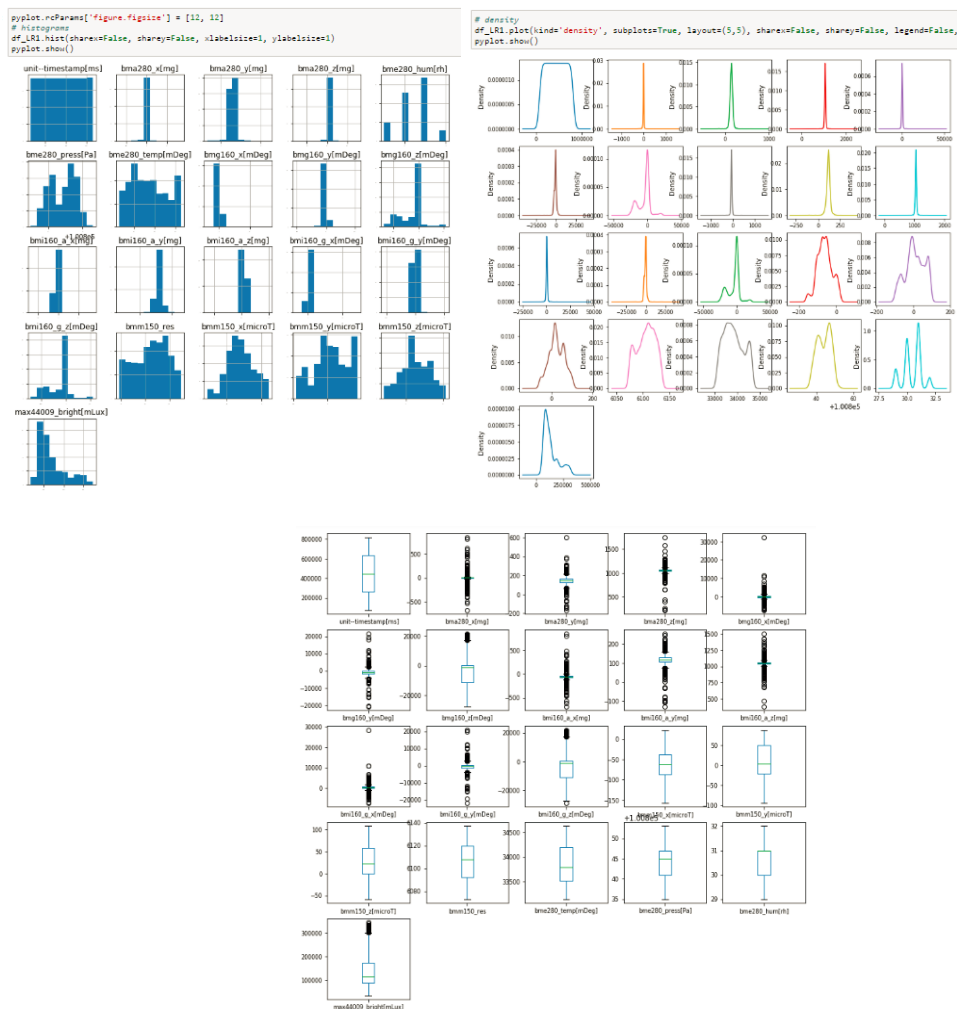


Figure 11 – Histogram/ Density/ Box Plots

4.1.3 Data Preparation

4.1.3.1 Feature Selection

The table below summarizes the features under consideration for selection:

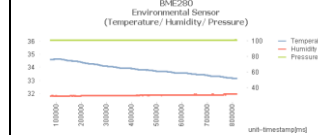
| Sensor | Feature Name | Remarks |
|--------|---|--|
| BMA280 | bma280_x, bma280_y, bma280_z | Duplicate measurements by both sensors. Only the BMI160 sensor measurements are considered |
| BMI160 | bmi160_a_x, bmi160_a_y, bmi160_a_z | |
| | bmi160_g_x, bmi160_g_y, bmi160_g_z | Duplicate measurements by both sensors. Only the BMI160 sensor measurements are considered |
| BMG160 | bmg160_x, bmg160_y, bmg160_z | |
| BME280 | bme280_temp (Temperature) bme280_hum (Humidity) bme280_press (Pressure) max44009_bright[mLux] (Brightness) |  <p>Temperature, Pressure and Humidity measurement show low correlation to cart movement i.e. not selected. Brightness is dependent on external factors e.g. lighting, time of day etc. and not reliable for clustering.</p> |
| BM150 | bmm150_x, bmm150_y, bmm150_z bmm150_res | Magnetic field measurements are location-dependent and not reliable for clustering. |

Table 7 – Feature Selection

Based on the above table, only the BMI160 measurements are considered. The table below shows the plot of the linear acceleration and angular velocity measurements from the BMI160 inertial measurement unit:

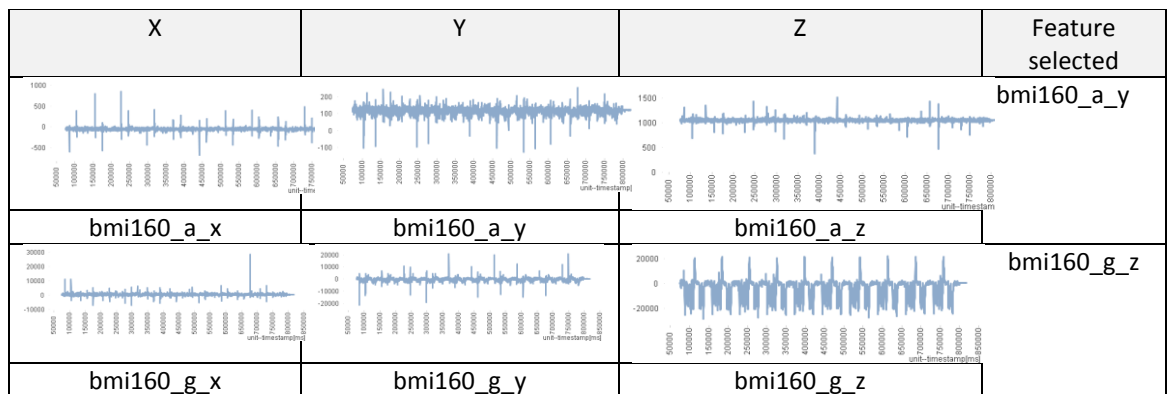


Table 8 – BMI160 measurements feature consideration

Based on the above table, only the “bmi160_a_y” and “bmi160_g_z” features show sufficient variation and trend for machine learning, and are selected.

4.1.4 Clustering Evaluation

4.1.4.1 Select Optimal K

The raw time series data collected are unlabelled and thus, unsupervised machine learning techniques are applicable. For the purpose of this project, clustering techniques are evaluated to select the most suitable to be used

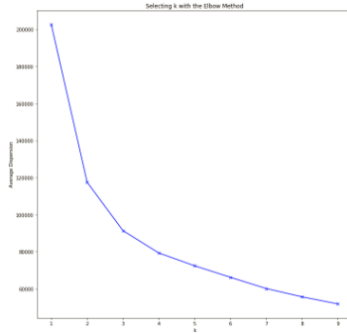


Figure 12 – Number of clusters K selection using Elbow Method

To determine the number of clusters to use, the elbow method is used. From the figure above, the optimal value for K where the improvements (centroids are closest to the clusters centroids) decline most rapidly is K=4. This is used to evaluate the various clustering algorithms used.

4.1.4.2 Evaluate Algorithm Performance

The figure below shows the scatter plot of Gyro Z-axis Vs Accelerometer Y-axis showing cluster identification using different learning algorithms:

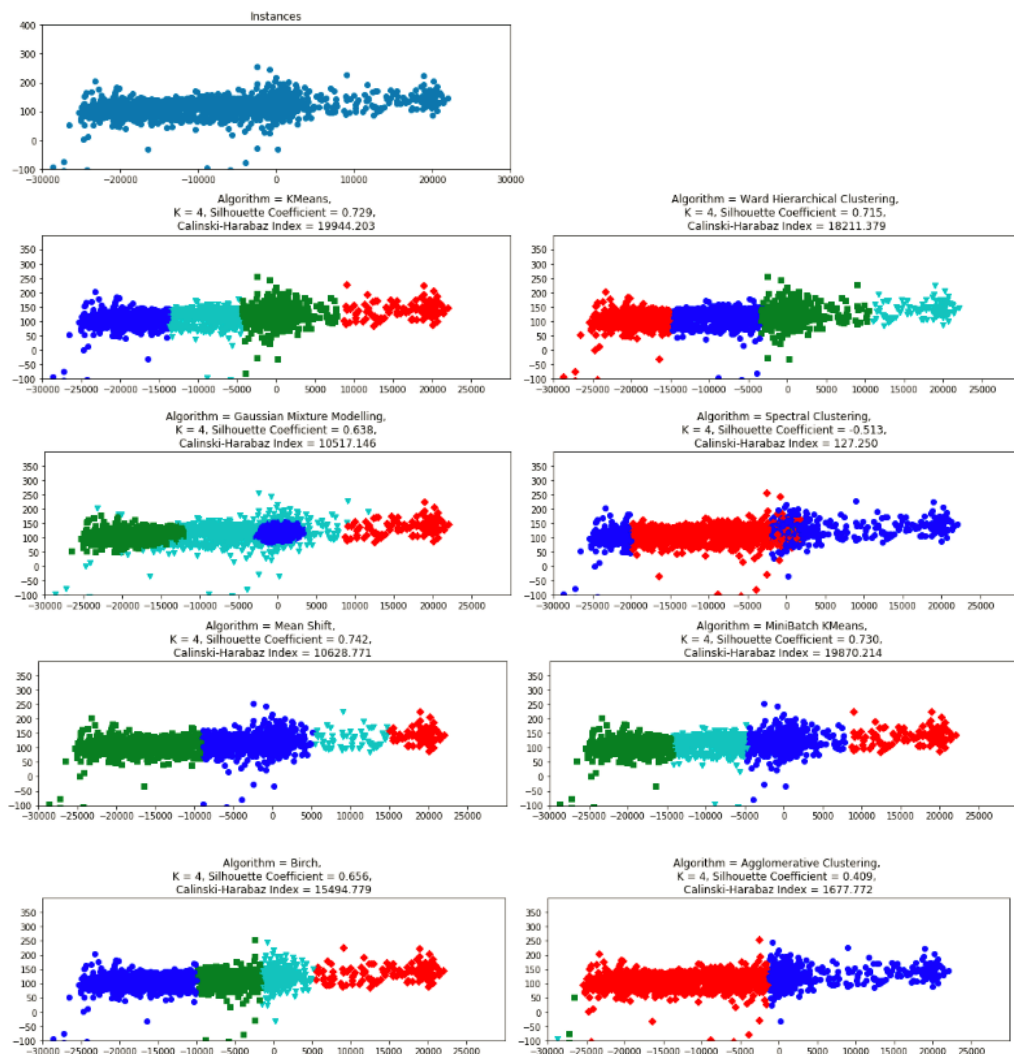


Figure 13 – Lap Run 1 Data Classification Algorithm Evaluation from scatter plots

The performance of the clustering results is measured using the Silhouette Coefficient (a number closer to 1 indicates better clustering result) and Calinski-Harabaz Index (higher value indicates dense and well separated clusters). From figure 13 above, it can be observed that KMeans, MiniBatch KMeans, Ward Hierarchical Clustering, and Mean Shift are considered for further evaluation.

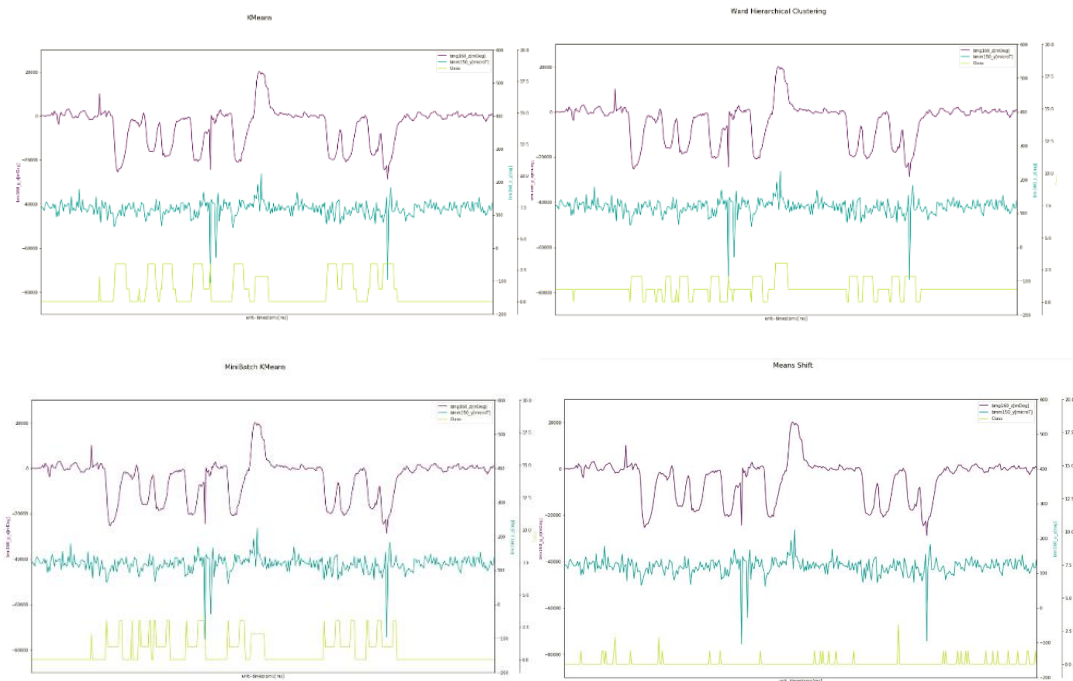
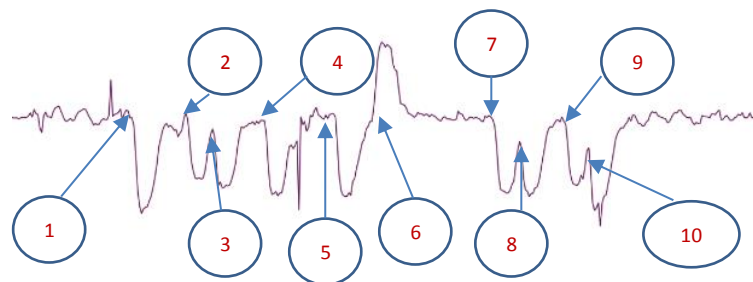


Figure 14 – Lap Run 1 Data Classification Algorithm Evaluation from time-series plots

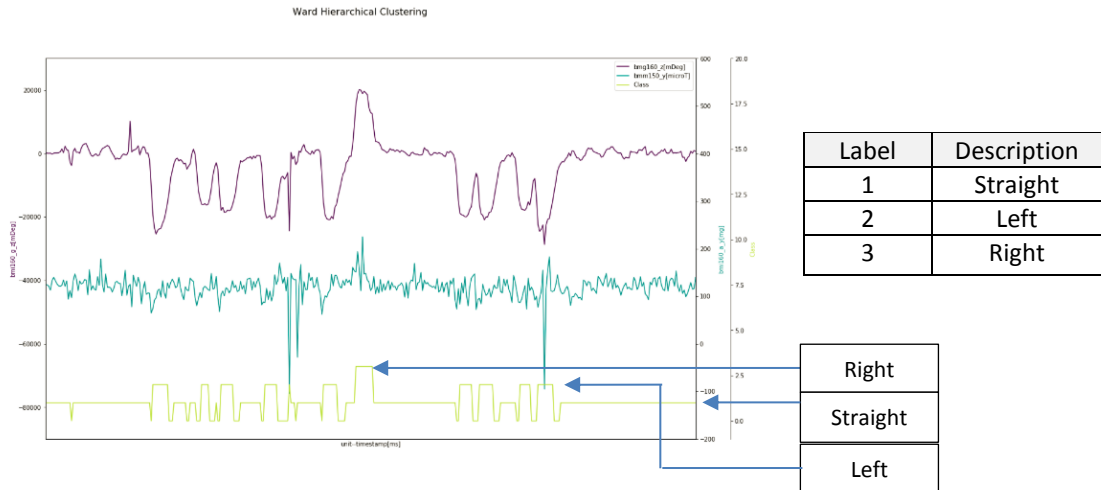
The performance of the clustering results is determined by how well the clustering results correlate to the segments derived from manual visual identification as below:



| | 1 -> 2 | 2 -> 3 | 3 -> 4 | 4 -> 5 | 5 -> 6 | 6 -> 7 | 7 -> 8 | 8 -> 9 | 9 -> 10 | 10 -> 1 |
|---------|--------|--------|--------|--------|--------|--------|--------|--------|---------|---------|
| Segment | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

Figure 15 – Segment Identification via visual scanning

Based on the above, the results using Ward Hierarchical Clustering show the clearest separation and identification of segments as compared to that from using the Birch algorithm which performed the poorest amongst the four.



4.1.4.3 Evaluate Algorithm Performance on Validation/ Test Dataset

The figure below shows the result of applying the clustering algorithm on the validation dataset from lap run 2:

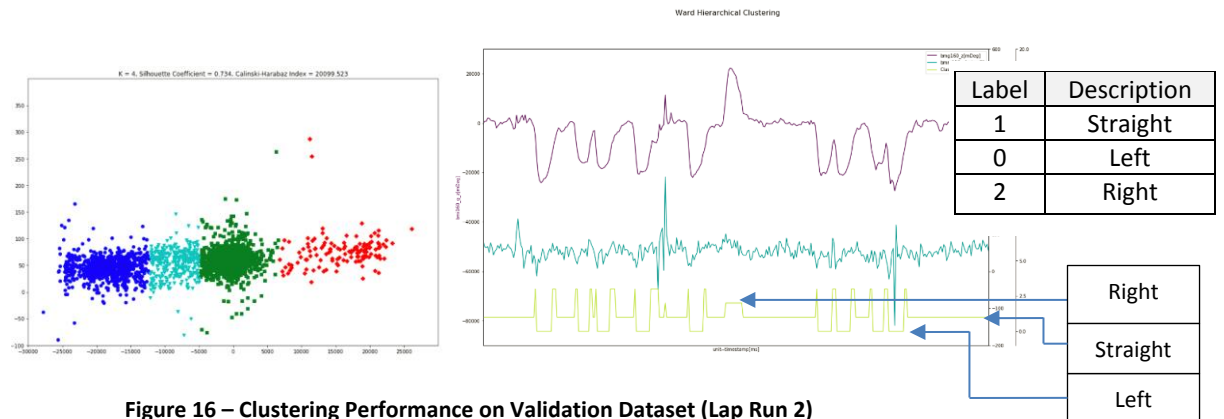


Figure 16 – Clustering Performance on Validation Dataset (Lap Run 2)

It is observed that the clustering performance is satisfactory except that the clustering labels for Left is 0 instead of 2 and Right is 2 instead of 3.

The figure below shows the result of applying the clustering algorithm on the validation dataset from lap run 3:

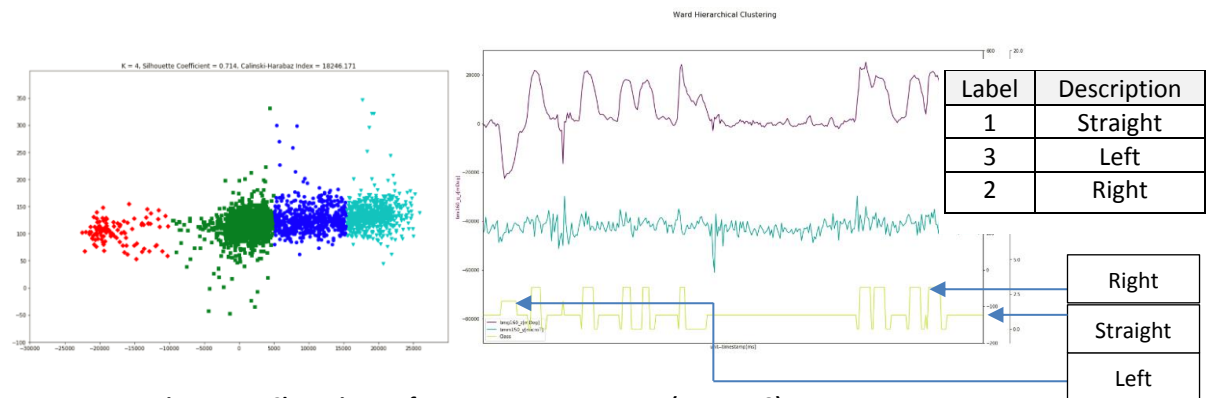


Figure 17 – Clustering Performance on Test Dataset (Lap Run 3)

It is observed that the clustering performance is satisfactory except that the clustering labels for Left is 2 instead of 3 and Right is 3 instead of 2.

In both cases above, further treatment (re-labelling is required before further classification is carried out and deriving the lap timings from the clustering results.

4.1.4.4 Lap Timing Derivation

Based on the output result from the Ward Hierarchical Clustering algorithm, the lap and individual segment timings for each lap can be derived:

| Lap | Lap Run 1 | Lap Run 2 | Lap Run 3 |
|--------------|-----------|-----------|-----------|
| 1 | 71 | 77.4 | 71.8 |
| 2 | 71 | 76.2 | 72.4 |
| 3 | 71.2 | 78.4 | 72.6 |
| 4 | 71.6 | 79.4 | 73 |
| 5 | 71.6 | 78.6 | 73 |
| 6 | 72.2 | 79.8 | 73.6 |
| 7 | 72 | 82.8 | 73.4 |
| 8 | 72.4 | 82.6 | 73 |
| 9 | 72.2 | 82.8 | 73.4 |
| Mean | 71.7 | 79.8 | 72.9 |
| Total (secs) | 645.2 | 718 | 656.2 |
| Total (mins) | 10.753 | 11.967 | 10.937 |

Table 9 – Lap Timings for Lap Run 1/2/3

From the above table, it can be observed that for normal condition in Lap Run 1 i.e. no obstacles, the average lap timing is about 72 seconds. This is fairly consistent with the average timing for Lap Run 3. It can be concluded that the average time to complete 1 lap is about 72 seconds.

In Lap Run 2, the lap timing is approximately 80s which is approximately 10% higher than normal. This is due to the frequent stop-starts caused by passing objects.

4.1.4.5 Overall Equipment Effectiveness (OEE) Calculation

Based on the lap timings derived in section 4.1.4.4, it is possible to derive the performance factor component required to calculate the OEE of the autonomous robotic cart.

| Support Variable | Calculation | Calculated Data | Result | Remarks |
|-------------------------|--|---------------------------------|---------------|--------------------------|
| Shift Length | - | - | 45 mins | 10:50 a.m. to 11:35 a.m. |
| Breaks | - | - | 0 mins | - |
| Setup/ Adjustment Time | Planned Production Time – Operating Time | 45 - (10.753 + 11.967 + 10.937) | 11.343 mins | |
| Run Rate | Total Laps / Operating Time | 27 / 33.657 | 0.802 per min | |
| Ideal Run Rate | Ideal Run Rate * | 1 / (72/60) | 0.833 per min | |
| Planned Production Time | Shift Length - Breaks | 45 – 0 mins | 45 mins | - |

| Support Variable | Calculation | Calculated Data | Result | Remarks |
|------------------------------|-------------------------------------|------------------|-------------|---------------------------------|
| Operating Time ** | Planned Production Time – Down Time | 45 - 11.343 mins | 33.657 mins | - |
| Defect Rate (Total Rejected) | - | - | 0 | All laps successfully completed |

Table 10 – Supporting Variables used for OEE calculation

The table below shows how the OEE percentage is calculated using the collected data and calculated support variables:

| OEE Factor | Calculation | Calculated Data | OEE % | World Class |
|---------------------|--|------------------|-----------------|-------------|
| Availability | Operating Time / Planned Production Time | 33.657 / 45 mins | 0.7479 (74.79%) | 90% |
| Performance | Run Rate / Ideal Run Rate | 0.802 / 0.833 | 0.9628 (96.28%) | 95% |
| Quality | Completed Laps / Planned Laps | 27 / 27 | 1 (100%) | 99.99% |
| Overall OEE | Availability x Performance x Quality | 0.8 x 0.9628 x 1 | 0.72 (72%) | 85% |

Table 11 – OEE Calculation

Note:

* Ideal Run Rate = Number of laps completed per minute (based on average lap timing of 72 seconds).

** Operating Time = Total Time for Lap Run 1 + Total Time for Lap Run 2 + Total Time for Lap Run 3

The final calculated OEE percentage is about 72%, which is still considerably lower than the World Class OEE benchmark of 85% or better.

From the table above, when comparing the individual OEE factors against the World Class benchmark, it is observed that the Availability factor is below the benchmark of 90% (highlighted in yellow). To improve this metric, it is proposed to implement measures to reduce the time for setup/ adjustments in between each Lap Run (current is 11.343 mins in total or 3.7 mins per Lap Run).

4.1.5 Time Series Analysis

4.1.5.1 Introduction

The output from the clustering operation allows the individual segment timings to be derived for each Lap Run, and this can be further analysed to provide alert notifications for any performance deviation.

4.1.5.2 SARIMA Analysis

The figure below shows the series and seasonal decomposition plot of the segment timings for Lap Run 1:

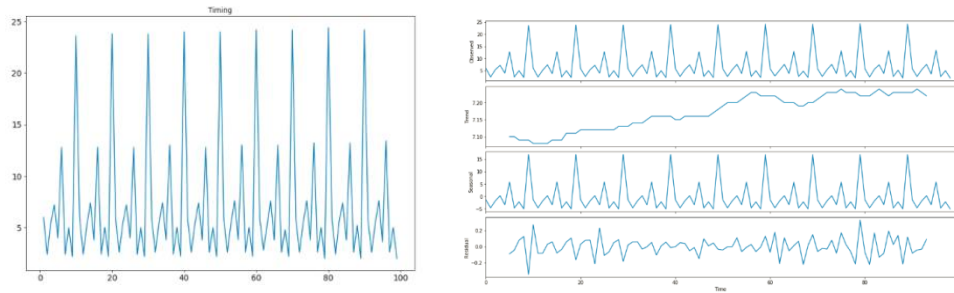


Figure 18 –Series and seasonal decomposition plot of Lap Run 1 segment timings

The plots above show that there is a strong seasonal component and slightly increasing trend for the segment timing series data.

Non-Differenced Data Result:
 ADF Statistic: -1.423053
 p-value: 0.571176
 Critical Values:
 1%: -3.509
 5%: -2.896
 10%: -2.585

Differenced Data Result:
 ADF Statistic: -14.683308
 p-value: 0.000000
 Critical Values:
 1%: -3.507
 5%: -2.895
 10%: -2.585

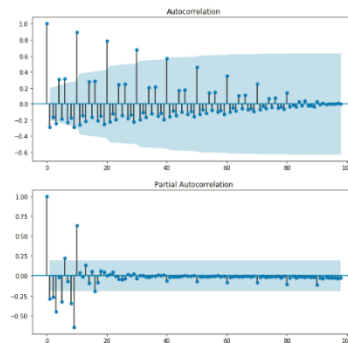


Figure 19 –ACF/ PACF Plots for Lap Run 1 segment timings

Based on the ACF and PACF plots, the first significant value is at lag 0 for ACF and for the PACF, thus $p = 0$ and $q = 0$. The large value at lag 10 in ACF plot suggests season $S = 10$ and since this lag is positive it suggests $P = 1$ and $Q = 0$. Since this is a differenced series for SARIMA $d = 1$, and since the seasonal pattern is not stable over time we set $D = 0$. Putting all these together gives a SARIMA(0,1,0)(1,0,0)[10] model. SARIMA analysis is performed using these values to fit a model on the segment timings for Lap Run 1.

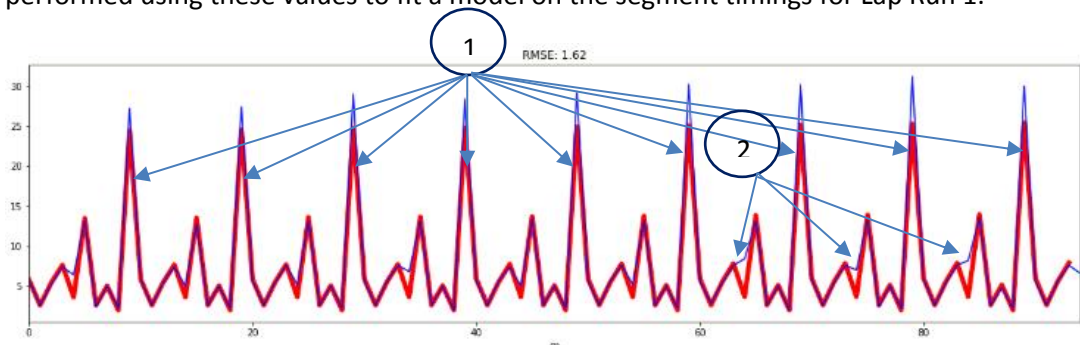


Figure 20 –Forecasted/ Lap Run 2 segment timing overlay plot

By plotting the forecasts generated from the SARIMA analysis and overlaying it with the segment timings from Lap Run 2, it is observed that the segment timings quite closely matches the generated forecasts, except for points 1 and 2.

These forecasts generated from SARIMA can be used to provide targeted notification of performance deviation via alerts e.g. through email, SMS etc. for necessary follow-up action.

4.1.6 Multi-Class Classification

4.1.6.1 Introduction

The output labels provided by the clustering operation on the data from Lap Run 1 can be used to train a machine learning model to provide automatic labelling e.g. Left, Right, Straight, etc. on data for subsequent lap runs. This is useful for providing real-time status of the autonomous robotic cart which can be accessed via remote monitoring displays.

4.1.6.2 Training/Validation Dataset and Evaluation Metric

80% of the data from Lap Run 1 is used to develop the training model and the remaining 20% of data is used to validate the developed model.

4.1.6.3 Spot Check Algorithms

We evaluate 6 classifier algorithms (Logistic Regression (LR), Linear Discriminant Analysis (LDA), Support Vector Classifier (SVC), K-Nearest Neighbour Classifier (KNN), Gaussian Naive Bayes (GaussianNB) and Decision Tree Classifier) using default tuning parameters with the following accuracy classification scores:

Algorithm results on raw data

```
LR: 0.890082 (0.024944)
LDA: 0.956098 (0.007420)
SVC: 0.618307 (0.028134)
KNN: 0.884045 (0.022487)
GaussianNB: 0.936325 (0.011990)
DecisionTree: 0.999666 (0.001003)
```

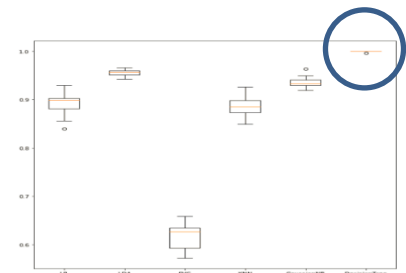


Figure 21 –Algorithm Spot Check Results

From the results above, it can be observed that Decision Tree Classifier is the outperformer as compared to the other algorithms.

4.1.6.4 Compare Algorithms

Next, we evaluate the effect of performing data transformations on the training data (standardization, normalization, scaling and binarization) on the performance of the 6 classifier algorithms and obtained the following accuracy classification scores:

Algorithm results on Standardized Data

```
LR: 0.858899 (0.032855)
LDA: 0.956098 (0.007420)
SVC: 0.962468 (0.011101)
KNN: 0.930302 (0.017797)
GaussianNB: 0.937331 (0.011890)
DecisionTree: 0.999666 (0.001003)
```

Algorithm results on Rescaled Data

```
LR: 0.832098 (0.030388)
LDA: 0.956098 (0.007420)
SVC: 0.908503 (0.021331)
KNN: 0.934652 (0.013501)
GaussianNB: 0.927321 (0.011800)
DecisionTree: 0.999666 (0.001003)
```

Algorithm results on Normalized Data

```
LR: 0.686000 (0.029940)
LDA: 0.853221 (0.023489)
SVC: 0.618307 (0.028134)
KNN: 0.917220 (0.019336)
GaussianNB: 0.872654 (0.029055)
DecisionTree: 0.965151 (0.015214)
```

Algorithm results on Binarized Data

```
LR: 0.722180 (0.030161)
LDA: 0.705774 (0.025327)
SVC: 0.716819 (0.026422)
KNN: 0.700095 (0.025997)
GaussianNB: 0.540216 (0.026118)
DecisionTree: 0.733583 (0.022360)
```

Figure 22 –Algorithm Comparison on Transformed Data Results

From the results above, it is observed that performing standardization on the training data produces better results for Support Vector and K-Nearest Neighbour Classifiers. However, Decision Tree Classifier still outperforms the rest.

4.1.6.5 *Estimate accuracy on validation dataset*

Next, we estimate the accuracy classification score metric of the Decision Tree Classifier using transformed data (standardization, normalization, scaling and binarization) on the validation dataset and obtained the following accuracy classification scores:

```
Accuracy Estimation of DecisionTree on validation dataset
Standardized: 0.9973226238286479
Normalized: 0.9585006693440429
Rescaled: 0.9049531459170014
Binarized: 0.7041499330655957
```

Figure 23 –Validation Dataset accuracy estimation results

From the results above, it is observed that the Decision Tree Classifier produces the best score with a standardized copy of the training dataset.

4.1.6.6 *Algorithm Tuning*

Next, we attempt to improve the result of the Decision Tree Classifier by using grid search to try a set of different max depth values from 1 to 12, with each value evaluated using 10-fold cross-validation on a standardized copy of the training dataset:

```
Best: 0.999665 using {'max_depth': 2}
0.820710 (0.025099) with: {'max_depth': 1}
0.999665 (0.001004) with: {'max_depth': 2}
0.999665 (0.001004) with: {'max_depth': 3}
0.999665 (0.001004) with: {'max_depth': 4}
0.999665 (0.001004) with: {'max_depth': 5}
0.999665 (0.001004) with: {'max_depth': 6}
0.999665 (0.001004) with: {'max_depth': 7}
0.999665 (0.001004) with: {'max_depth': 8}
0.999665 (0.001004) with: {'max_depth': 9}
0.999665 (0.001004) with: {'max_depth': 10}
0.999665 (0.001004) with: {'max_depth': 11}
0.999665 (0.001004) with: {'max_depth': 12}
```

Figure 24 –Algorithm Tuning on Decision Tree Classifier Results

From the results above, the best value for “max_depth” parameter for the Decision Tree Classifier is 2, providing an accuracy classification score of 99.966%.

4.1.6.7 *Ensemble Methods*

4.1.6.7.1 *Spot Check Algorithms*

Next, we attempt to use ensemble methods (AdaBoost, GradientBoosting, RandomForest, ExtraTrees, Bagging and Voting) to improve the algorithm

performance using default tuning parameters which produces the following accuracy classification scores:

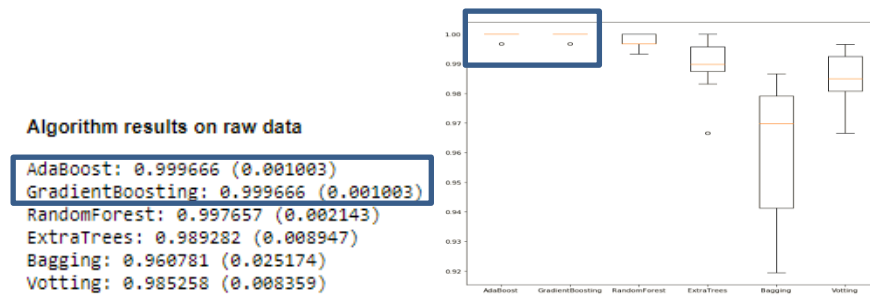


Figure 25 –Ensemble Methods Algorithm Spot Check Results

From the results above, it can be observed that AdaBoost and Gradient Boosting Classifiers are the outperformer as compared to the other algorithms.

4.1.6.7.2 Compare Algorithms

Next, we evaluate the effect of performing data transformations on the training data (standardization, normalization, scaling and binarization) on the performance of the 6 ensemble methods and obtained the following accuracy classification scores:

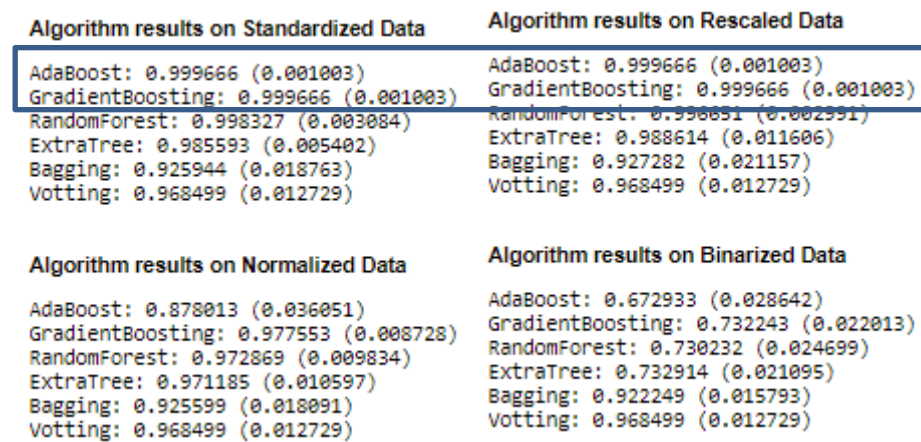


Figure 26 – Ensemble Methods Algorithm Comparison on Transformed Data Results

From the results above, it is observed that AdaBoost and GradientBoosting are the outperformers producing the best accuracy classification scores.

4.1.6.7.3 Estimate accuracy on validation dataset

Next, we estimate the accuracy classification score metric of AdaBoost and GradientBoosting classifiers using transformed data (standardization, normalization, scaling and binarization) on the validation dataset and obtained the following accuracy classification scores:

Accuracy Estimation of AdaBoost on validation dataset

Standardized: 0.9973226238286479
Normalized: 0.9089092101740294
Rescaled: 0.9049531459170014
Binarized: 0.6519410977242303

Accuracy Estimation of GradientBoosting on validation dataset

Standardized: 0.9973226238286479
Normalized: 0.9772423023435074
Rescaled: 0.9049531459170014
Binarized: 0.7081659973226239

Figure 27 –Ensemble Method Validation Dataset accuracy estimation results

From the results above, it is observed that AdaBoost and Gradient Boosting produce the best score amongst all others with a standardized copy of the training dataset.

4.1.6.7.4 Algorithm Tuning

Next, we attempt to improve the result of AdaBoost Classifier by using grid search to try a set of different “n_estimators” and “learning_rate” values from 50 to 100 and 0.1 to 2.0 respectively, with each value evaluated using 10-fold cross-validation on a standardized copy of the training dataset:

```
Best: 0.999665 using {'learning_rate': 0.1, 'n_estimators': 90}
0.951743 (0.057367) with: {'learning_rate': 0.1, 'n_estimators': 50}
0.990282 (0.013342) with: {'learning_rate': 0.1, 'n_estimators': 60}
0.997319 (0.006989) with: {'learning_rate': 0.1, 'n_estimators': 70}
0.997319 (0.006989) with: {'learning_rate': 0.1, 'n_estimators': 80}
0.999665 (0.001004) with: {'learning_rate': 0.1, 'n_estimators': 90}
0.999665 (0.001004) with: {'learning_rate': 0.1, 'n_estimators': 100}
0.999665 (0.001004) with: {'learning_rate': 0.5, 'n_estimators': 50}
0.999665 (0.001004) with: {'learning_rate': 0.5, 'n_estimators': 60}
0.999665 (0.001004) with: {'learning_rate': 0.5, 'n_estimators': 70}
0.999665 (0.001004) with: {'learning_rate': 0.5, 'n_estimators': 80}
0.999665 (0.001004) with: {'learning_rate': 0.5, 'n_estimators': 90}
0.999665 (0.001004) with: {'learning_rate': 0.5, 'n_estimators': 100}
0.999665 (0.001004) with: {'learning_rate': 1.0, 'n_estimators': 50}
0.999665 (0.001004) with: {'learning_rate': 1.0, 'n_estimators': 60}
0.999665 (0.001004) with: {'learning_rate': 1.0, 'n_estimators': 70}
0.999665 (0.001004) with: {'learning_rate': 1.0, 'n_estimators': 80}
0.999665 (0.001004) with: {'learning_rate': 1.0, 'n_estimators': 90}
0.999665 (0.001004) with: {'learning_rate': 1.0, 'n_estimators': 100}
0.806300 (0.023168) with: {'learning_rate': 2.0, 'n_estimators': 50}
0.808646 (0.025705) with: {'learning_rate': 2.0, 'n_estimators': 60}
0.816354 (0.029058) with: {'learning_rate': 2.0, 'n_estimators': 70}
0.822721 (0.027248) with: {'learning_rate': 2.0, 'n_estimators': 80}
0.828418 (0.024655) with: {'learning_rate': 2.0, 'n_estimators': 90}
0.828418 (0.024655) with: {'learning_rate': 2.0, 'n_estimators': 100}
```

Figure 28 –Algorithm Tuning on AdaBoost Classifier Results

From the results above, the best combination for “n_estimators” and “learning_rate” parameters for the AdaBoost Classifier is 90 and 0.1 respectively, providing an accuracy classification score of 99.966%.

Similarly, we attempt to improve the result of Gradient Boosting Classifier by using grid search to try a set of different “n_estimators” values from 50 to 400, with each value evaluated using 10-fold cross-validation on a standardized copy of the training dataset:

```
Best: 0.999665 using {'n_estimators': 50}
0.999665 (0.001004) with: {'n_estimators': 50}
0.999665 (0.001004) with: {'n_estimators': 100}
0.999665 (0.001004) with: {'n_estimators': 150}
0.999665 (0.001004) with: {'n_estimators': 200}
0.999665 (0.001004) with: {'n_estimators': 250}
0.999665 (0.001004) with: {'n_estimators': 300}
0.999665 (0.001004) with: {'n_estimators': 350}
0.999665 (0.001004) with: {'n_estimators': 400}
```

Figure 29 –Algorithm Tuning on Gradient Boosting Classifier Results

From the results above, the best value for “n_estimators” parameter for the Gradient Boosting Classifier is 50, providing an accuracy classification score of 99.966%.

4.1.6.8 Finalize Model

4.1.6.8.1 Prepare/ Save/ Load/ Test Model

Based on the results obtained from the classification analysis in the previous sections, Decision Tree, AdaBoost and Gradient Boosting Classifiers produces the best accuracy estimation scores amongst the 12 algorithms (6 classification and 6 ensemble methods) evaluated using a standardized copy of the dataset.

As the accuracy performance is similar using the 3 algorithms, the AdaBoost Classifier is selected for use to test predictions on the dataset from Lap Run 2.

Accuracy Classification Score:

0.9403312226010716

Confusion Matrix:

```
[[ 352    0    0    0]
 [ 111 2685    0    0]
 [ 115    0  712    0]
 [    0   19    0 112]]
```

Classification Report:

| | precision | recall | f1-score | support |
|-------------|-----------|--------|----------|---------|
| 0 | 0.61 | 1.00 | 0.76 | 352 |
| 1 | 0.99 | 0.96 | 0.98 | 2796 |
| 2 | 1.00 | 0.86 | 0.93 | 827 |
| 3 | 1.00 | 0.85 | 0.92 | 131 |
| avg / total | 0.96 | 0.94 | 0.95 | 4106 |

Figure 30 –Prediction Result on Test Dataset

From the results above, the overall accuracy classification score is 94.03% which is quite good. The precision scores of 1 for categories 2 (left) and 3 (right) shows that the algorithm was able to correctly classify all samples belonging to these categories.

4.1.6.9 Summary

In this section, various learning algorithms were evaluated to perform multi-class classification on the time series data from the autonomous robotic cart. AdaBoost Classifier was eventually selected as the learning model to be used for new data classification. Tuning was performed using Grid Search to select the optimal combination of parameters to produce the best accuracy estimation result.

5. Reflections

5.1 Challenges/ Recommendations

1. The task of clustering the time series data was a non-trivial task and involves a fair amount of trial-and-error in order to select the most appropriate learning algorithm. Even so, the developed model is valid only if the data collected is representative of all future movements that the autonomous robotic cart will make as it traverses through its designated path of travel. Any movement changes might require retraining of the learning model with new data to ensure it is still relevant and applicable.
2. The analysis has assumed that the autonomous robotic cart is travelling on an even terrain, and thus the model developed does not account for unexpected obstacles that could cause jerky movement or toppling over in the worst case scenario. To incorporate such anomalies would require additional mechanism to be developed for anomaly detection, and Scikit-Learn does have some learning models e.g. One-Class SVM, Local Outlier Factor etc. that can potentially be used to develop the appropriate model.
3. The XDK110 sensor device normally runs on a Lithium-Ion rechargeable battery and thus, battery life needs to be periodically monitored physically to ensure the device is still operational. Further enhancements could include incorporating device battery status monitoring which could be used to provide alerts on the ThingWorX IoT platform.

6. Conclusion

In this report, an IoT solution was designed for the Thouzer autonomous robotic cart using the Bosch XDK110 sensor device to collect telemetry data and using the PTC ThingWorX IoT platform for real-time telemetry data monitoring and visualization. The collected data was analysed using machine learning techniques and OEE calculation methodology was developed to monitor the productivity of the robotic cart.

7. References

1. PTC University Precision LMS (<https://precisionlms.ptc.com/app/home?1>)
2. Doog-inc Website (<http://jp.doog-inc.com/product-thouzer.html>)
3. XDK Cross Domain Development Kit Website (<https://xdk.bosch-connectivity.com>)
4. Scikit-Learn Clustering Documentation
(<http://scikit-learn.org/stable/modules/clustering.html#clustering>)
5. Scikit-Learn Ensemble Methods Documentation
(<http://scikit-learn.org/stable/modules/ensemble.html>)
6. Scikit-Learn Multi-class Algorithms Documentation
(<http://scikit-learn.org/stable/modules/multiclass.html>)

8. Appendix 1 – Source Code Listing

8.1 XDK Workbench

Main.c

```
/*
 * Licensee agrees that the example code provided to Licensee has been developed and released by Bosch
 * solely as an example to be used as a potential reference for Licensee's application development.
 * Fitness and suitability of the example code for any use within Licensee's applications need to be verified by
 * Licensee on its own authority by taking appropriate state of the art actions and measures (e.g. by means of
 * quality assurance measures).
 * Licensee shall be responsible for conducting the development of its applications as well as integration of
 * parts of the example code into such applications, taking into account the state of the art of technology and any
 * statutory regulations and provisions applicable for such applications. Compliance with the functional system
 * requirements and testing there of (including validation of information/data security aspects and functional
 * safety) and release shall be solely incumbent upon Licensee.
 * For the avoidance of doubt, Licensee shall be responsible and fully liable for the applications and any
 * distribution of such applications into the market.
 *
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are
 * met:
 *
 * (1) Redistributions of source code must retain the above copyright
 * notice, this list of conditions and the following disclaimer.
 *
 * (2) Redistributions in binary form must reproduce the above copyright
 * notice, this list of conditions and the following disclaimer in
 * the documentation and/or other materials provided with the
 * distribution.
 *
 * (3)The name of the author may not be used to
 * endorse or promote products derived from this software without
 * specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS" AND ANY EXPRESS OR
 * IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
 * DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT,
 * INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
 * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGE.
 */

/* system header files */
#include <stdio.h>
#include "BCDS_Basics.h"

/* additional interface header files */
```

```

#include "XdkSystemStartup.h"
#include "BCDS_Assert.h"
#include "BCDS_CmdProcessor.h"
#include "FreeRTOS.h"
#include "task.h"
#include "BOSCH_THINGWORX.h"

/* own header files */

/* global variables *****/
static CmdProcessor_T MainCmdProcessor;

/* functions */

int main(void)
{
    /* Mapping Default Error Handling function */
    Retcode_T returnValue = Retcode_Initialize(DefaultErrorHandlingFunc);
    if (RETCODE_OK == returnValue)
    {
        returnValue = systemStartup();
    }
    if (RETCODE_OK == returnValue)
    {
        returnValue = CmdProcessor_initialize(&MainCmdProcessor, (char *) "MainCmdProcessor",
        TASK_PRIO_MAIN_CMD_PROCESSOR, TASK_STACK_SIZE_MAIN_CMD_PROCESSOR,
        TASK_Q_LEN_MAIN_CMD_PROCESSOR);
    }
    if (RETCODE_OK == returnValue)
    {
        /* Here we enqueue the application initialization into the command
        * processor, such that the initialization function will be invoked
        * once the RTOS scheduler is started below.
        */
        returnValue = CmdProcessor_enqueue(&MainCmdProcessor, applInitSystem, &MainCmdProcessor,
        UINT32_C(0));
    }
    if (RETCODE_OK != returnValue)
    {
        printf("System Startup failed");
        assert(false);
    }
    /* start scheduler */
    vTaskStartScheduler();
}

```

BOSCH THINGWORX.c

/*

* Licensee agrees that the example code provided to Licensee has been developed and released by Bosch solely as an example to be used as a potential reference for Licensee's application development.

* Fitness and suitability of the example code for any use within Licensee's applications need to be verified by Licensee on its own authority by taking appropriate state of the art actions and measures (e.g. by means of quality assurance measures).

* Licensee shall be responsible for conducting the development of its applications as well as integration of parts of the example code into such applications, taking into account the state of the art of technology and any statutory regulations and provisions applicable for such applications. Compliance with the functional system requirements and testing thereof (including validation of information/data security aspects and functional safety) and release shall be solely incumbent upon Licensee.

* For the avoidance of doubt, Licensee shall be responsible and fully liable for the applications and any distribution of such applications into the market.

*

*

* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions are
* met:

*

* (1) Redistributions of source code must retain the above copyright
* notice, this list of conditions and the following disclaimer.

*

* (2) Redistributions in binary form must reproduce the above copyright
* notice, this list of conditions and the following disclaimer in
* the documentation and/or other materials provided with the
* distribution.

*

* (3) The name of the author may not be used to
* endorse or promote products derived from this software without
* specific prior written permission.

*

* THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS" AND ANY EXPRESS OR
* IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
* WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
* DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT,
* INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
* (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
* SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
* HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
* STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
* IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
* POSSIBILITY OF SUCH DAMAGE.

*/

/*-----*/

/**

* @ingroup APPS_LIST

*

* @defgroup XDK_APPLICATION_TEMPLATE XDK Application Template

* @{

*

* @brief XDK Application Template

*

* @details Empty XDK Application Template without any functionality. Should be used as a template to start new projects.

*

* @file

**/

/* module includes *****/

/* system header files */

#include <stdio.h>

/* additional interface header files */

#include "FreeRTOS.h"

#include "timers.h"

#include <Serval_RestClient.h>

/* own header files */

#include "BOSCH_THINGWORX.h"

#include "BCDS_CmdProcessor.h"

#include "BCDS_Assert.h"

#include "BCDS_WlanConnect.h"

#include "BCDS_NetworkConfig.h"

#include "PAL_initialize_ih.h"

#include "PAL_socketMonitor_ih.h"

/* Interface for all sensors on the XDK */

#include "XdkSensorHandle.h"

/* Sensor Handler for the BME280 Sensor */

extern Environmental_HandlePtr_T xdkEnvironmental_BME280_Handle;

extern Accelerometer_HandlePtr_T xdkAccelerometers_BMI160_Handle;

extern Gyroscope_HandlePtr_T xdkGyroscope_BMI160_Handle;

extern Magnetometer_HandlePtr_T xdkMagnetometer_BMM150_Handle;

/* constant definitions *****/

/* local variables *****/

/* global variables *****/

/* inline functions *****/

/* local functions *****/

static void onRESTRequestSent(Msg_T *msg_ptr, retcode_t status)

{

 (void) msg_ptr;

 if (status != RC_OK) {

 printf("Failed to send REST request!\r\n");

 }

}

static retcode_t onRESTResponseReceived(RestSession_T *restSession,

Msg_T *msg_ptr, retcode_t status)

{

 (void) restSession;

 if (status == RC_OK && msg_ptr != NULL) {

 Http_StatusCode_T statusCode = HttpMsg_getStatusCode(msg_ptr);

 char const *contentType = HttpMsg_getContentType(msg_ptr);

 char const *content_ptr;

```

        unsigned int contentLength = 0;
        HttpMsg_getContent(msg_ptr, &content_ptr, &contentLength);
        char content[contentLength+1];
        strncpy(content, content_ptr, contentLength);
        content[contentLength] = 0;
        printf("HTTP RESPONSE: %d [%s]\r\n", statusCode, contentType);
        printf("%s\r\n", content);
    } else {
        printf("Failed to receive REST response!\r\n");
    }
    return(RC_OK);
}

void StreamData(float value, char key[]){
    RestClient_initialize();

    /* Set IP Address and Port */
    Ip_Address_T destAddr;
    //Ip_convertOctetsToAddr(103, 57, 189, 187, &destAddr);
    //The library adds "http://" to the address automatically
    PAL_getIpAddress((uint8_t*) "rpthingworx.com", &destAddr);
    Ip_Port_T port = Ip_convertIntToPort(8080);

    Rest_ContentFormat_T acceptedFormats[1] = { REST_CONTENT_FMT_JSON };
    RestClient_ReqInfo_T putRequest;
    char payload[43];
    sprintf(payload, "{\"%s\": %f}", key, value);
    //printf("{\"Temperature\": %f}", number);
    putRequest.method = REST_PUT;
    putRequest.uriPath_ptr = "/Thingworx/Things/Bosch_LYH/Properties/*";
    putRequest.uriQuery_ptr = "?appKey=03c12ad1-8e7c-4d68-bdd8-49021dc94a94";
    putRequest.acceptBuffer_ptr = acceptedFormats;
    putRequest.numAccept = 1;
    putRequest.payloadLen = 43;
    putRequest.payload_ptr = (uint8_t*) payload;
    putRequest.contentType = REST_CONTENT_FMT_JSON;
    putRequest.rangeLength = 0;
    putRequest.host = NULL;

    Msg_T* msg_ptr;
    RestClient_initReqMsg(&destAddr, port, &msg_ptr, REST_CLIENT_PROTOCOL_HTTP);
    RestClient_fillReqMsg(msg_ptr, &putRequest);

    // send the request
    RestClient_request(msg_ptr, &onRESTRequestSent, &onRESTResponseReceived);
}

void initializeEnvironmentalSensor(){
    /* initialize environmental sensor */
    Retcode_T returnValue = RETCODE_FAILURE;
    returnValue = Environmental_init(xdkEnvironmental_BME280_Handle);

    /* initialize accelerometer sensor */
    Retcode_T AccelReturnValue = RETCODE_FAILURE;
    AccelReturnValue = Accelerometer_init(xdkAccelerometers_BMI160_Handle);
    if ( RETCODE_OK != AccelReturnValue) { // do something

```

```

        printf("Initialization Error for Accelerometer sensor\n");
    }

    /* initialize gyroscope sensor */
    Retcode_T gyroscopeInitReturnValue = RETCODE_FAILURE;
    gyroscopeInitReturnValue = Gyroscope_init(xdkGyroscope_BMI160_Handle);
    if (RETCODE_OK != gyroscopeInitReturnValue) { // do something
        printf("Initialization Error for Gyroscope sensor\n");
    }

    /* initialize magnetometer sensor */
    Retcode_T magnetometerInitReturnValue = RETCODE_FAILURE;
    magnetometerInitReturnValue = Magnetometer_init(xdkMagnetometer_BMM150_Handle);
    if (RETCODE_OK != magnetometerInitReturnValue) { // do something
        printf("Initialization Error for Magnetometer sensor\n");
    }

    if ( RETCODE_OK != returnValue) {
        printf("Initialization Error\n");
    }
}

void readEnvironmentalSensor(float* pressure, float* temperature, float* humidity){
    /* initialize environmental sensor */
    Retcode_T returnValue = RETCODE_FAILURE;
    returnValue = Environmental_init(xdkEnvironmental_BME280_Handle);
    if ( RETCODE_OK != returnValue) {
        // do something
    }
    /* read and print BME280 environmental sensor data */
    Environmental_Data_T bme280 = { INT32_C(0), UINT32_C(0), UINT32_C(0) };
    returnValue = Environmental_readData(xdkEnvironmental_BME280_Handle, &bme280);
    if ( RETCODE_OK == returnValue) {
        //printf("BME280 Environmental Conversion Data :\n\rp =%ld Pa\n\r t =%ld mDeg\n\r h
        =%ld %%rh\n\r", (long int) bme280.pressure, (long int) bme280.temperature, (long int) bme280.humidity);
        *pressure = (float) bme280.pressure;
        *temperature = (float) bme280.temperature / 1000;
        *humidity = (float) bme280.humidity;
    }
}

void readAccBMI160(int* AccX, int* AccY, int* AccZ){
    Retcode_T returnValue = RETCODE_FAILURE;
    /* read and print BMI160 accelerometer data */
    Accelerometer_XyzData_T bmiA160 = {INT32_C(0), INT32_C(0), INT32_C(0)};
    memset(&bmiA160, 0, sizeof(Accelerometer_XyzData_T));
    returnValue = Accelerometer_readXYZGValue(xdkAccelerometers_BMI160_Handle,&bmiA160);
    if (RETCODE_OK == returnValue){
        printf("BMI160 Acceleration Data \n\r: %f \n\r %f \n\r %f \n\r", (float) bmiA160.xAxisData,
        (float) bmiA160.yAxisData, (float) bmiA160.zAxisData);
        *AccX = (long int) bmiA160.xAxisData;
        *AccY = (long int) bmiA160.yAxisData;
        *AccZ = (long int) bmiA160.zAxisData;
    }
}

```



```

void readGyroBMI160(int* GyroX, int* GyroY, int* GyroZ){
    Retcode_T returnValue = RETCODE_FAILURE;
    /* read and print BMI160 gyroscope data */
    Gyroscope_XyzData_T bmiG160 = {INT32_C(0), INT32_C(0), INT32_C(0)};
    memset(&bmiG160, 0, sizeof(Gyroscope_XyzData_T));
    returnValue = Gyroscope_readXyzDegreeValue(xdkGyroscope_BMI160_Handle, &bmiG160);
    if (RETCODE_OK == returnValue){
        printf("BMI160 Gyro Data :\n\rx =%ld mDeg\n\r y =%ld mDeg\n\r z =%ld mDeg\n\r", (long int)
bmiG160.xAxisData, (long int) bmiG160.yAxisData, (long int) bmiG160.zAxisData);
        *GyroX = (long int) bmiG160.xAxisData;
        *GyroY = (long int) bmiG160.yAxisData;
        *GyroZ = (long int) bmiG160.zAxisData;
    }
}

void readMagBMM150(int* MagX, int* MagY, int* MagZ){
    Retcode_T returnValue = RETCODE_FAILURE;
    /* read and print BMM150 magnetometer data */
    Magnetometer_XyzData_T bmm150 = {INT32_C(0), INT32_C(0), INT32_C(0), INT32_C(0)};
    memset(&bmm150, 0, sizeof(Magnetometer_XyzData_T));

    returnValue = Magnetometer_readXyzTeslaData(xdkMagnetometer_BMM150_Handle, &bmm150);

    if (RETCODE_OK == returnValue) {
        printf("BMM150 Magnetic Data :\n\r x =%ld mT\n\r y =%ld mT\n\r z =%ld mT\n\r", (long int)
bmm150.xAxisData, (long int) bmm150.yAxisData, (long int) bmm150.zAxisData);
        *MagX = (long int) bmm150.xAxisData;
        *MagY = (long int) bmm150.yAxisData;
        *MagZ = (long int) bmm150.zAxisData;
    }
}

/* global functions ***** */

/**
 * Created By : Aldi Faizal, Mehul Gupta
 * Property and Copyright of CAD-IT CONSULTANTS (c) 2018
 *
 */
void appInitSystem(void * CmdProcessorHandle, uint32_t param2)
{
    if (CmdProcessorHandle == NULL)
    {
        printf("Command processor handle is null \n\r");
        assert(false);
    }
    BCDS_UNUSED(param2);

    /* Connect SDK to WiFi Network */

    WlanConnect_SSID_T connectSSID = (WlanConnect_SSID_T) "LYH_AndroidAP";
    WlanConnect_PassPhrase_T connectPassPhrase = (WlanConnect_PassPhrase_T) "fqgg68rq";
    WlanConnect_Init();
    NetworkConfig_SetIpDhcp(0);
    WlanConnect_WPA(connectSSID, connectPassPhrase, NULL);
    PAL_initialize();
    PAL_socketMonitorInit();

```

```

initializeEnvironmentalSensor();
float sensor_pressure, sensor_temperature, sensor_humidity;
int Acc_X, Acc_Y, Acc_Z;
int Gyro_X, Gyro_Y, Gyro_Z;
int Mag_X, Mag_Y, Mag_Z;

// Loop for sensor readings
for(;;){
    readEnvironmentalSensor(&sensor_pressure, &sensor_temperature, &sensor_humidity);
    StreamData(sensor_pressure, "Pressure");
    vTaskDelay((portTickType) 2*1000 / portTICK_RATE_MS);
    StreamData(sensor_temperature, "Temperature");
    vTaskDelay((portTickType) 2*1000 / portTICK_RATE_MS);
    StreamData(sensor_humidity, "Humidity");
    vTaskDelay((portTickType) 2*1000 / portTICK_RATE_MS);
    vTaskDelay((portTickType) 60*1000 / portTICK_RATE_MS);

    readAccBMI160(&Acc_X, &Acc_Y, &Acc_Z);
    StreamData(Acc_X, "Acc_X");
    //vTaskDelay((portTickType) 1*1000 / portTICK_RATE_MS);
    StreamData(Acc_Y, "Acc_Y");
    //vTaskDelay((portTickType) 1*1000 / portTICK_RATE_MS);
    StreamData(Acc_Z, "Acc_Z");
    //vTaskDelay((portTickType) 1*1000 / portTICK_RATE_MS);

    readGyroBMI160(&Gyro_X, &Gyro_Y, &Gyro_Z);
    StreamData(Gyro_X, "Gyro_X");
    //vTaskDelay((portTickType) 1*1000 / portTICK_RATE_MS);
    StreamData(Gyro_Y, "Gyro_Y");
    //vTaskDelay((portTickType) 1*1000 / portTICK_RATE_MS);
    StreamData(Gyro_Z, "Gyro_Z");
    //vTaskDelay((portTickType) 1*1000 / portTICK_RATE_MS);

    readMagBMM150(&Mag_X, &Mag_Y, &Mag_Z);
    StreamData(Mag_X, "Mag_X");
    //vTaskDelay((portTickType) 1*1000 / portTICK_RATE_MS);
    StreamData(Mag_Y, "Mag_Y");
    //vTaskDelay((portTickType) 1*1000 / portTICK_RATE_MS);
    StreamData(Mag_Z, "Mag_Z");
    //vTaskDelay((portTickType) 1*1000 / portTICK_RATE_MS);
}
}
/**@} */
/** ***** */

```

BOSCH THINGWORX.h

```
/*
 * Licensee agrees that the example code provided to Licensee has been developed and released by Bosch
solely as an example to be used as a potential reference for Licensee's application development.
 * Fitness and suitability of the example code for any use within Licensee's applications need to be verified by
Licensee on its own authority by taking appropriate state of the art actions and measures (e.g. by means of
quality assurance measures).
 * Licensee shall be responsible for conducting the development of its applications as well as integration of
parts of the example code into such applications, taking into account the state of the art of technology and any
statutory regulations and provisions applicable for such applications. Compliance with the functional system
requirements and testing there of (including validation of information/data security aspects and functional
safety) and release shall be solely incumbent upon Licensee.
 * For the avoidance of doubt, Licensee shall be responsible and fully liable for the applications and any
distribution of such applications into the market.
 *
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions are
 * met:
 *
 * (1) Redistributions of source code must retain the above copyright
 * notice, this list of conditions and the following disclaimer.
 *
 * (2) Redistributions in binary form must reproduce the above copyright
 * notice, this list of conditions and the following disclaimer in
 * the documentation and/or other materials provided with the
 * distribution.
 *
 * (3) The name of the author may not be used to
 * endorse or promote products derived from this software without
 * specific prior written permission.
 *
 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS" AND ANY EXPRESS OR
 * IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
 * WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
 * DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT,
 * INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
 * (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
 * STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING
 * IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 * POSSIBILITY OF SUCH DAMAGE.
 */
/*-----*/
/**
 * @file
 * @brief This Module is Configuration header for XDK Application Template configurations
 *
 */

/* header definition ***** */
#ifndef XDK110_XDKAPPLICATIONTEMPLATE_H_
#define XDK110_XDKAPPLICATIONTEMPLATE_H_
```

```

/* local interface declaration *****/
/* Priorities */
#define TASK_PRIO_MAIN_CMD_PROCESSOR      (UINT32_C(1))
#define TASK_STACK_SIZE_MAIN_CMD_PROCESSOR (UINT16_C(700))
#define TASK_Q_LEN_MAIN_CMD_PROCESSOR      (UINT32_C(10))
/* local type and macro definitions */

/* local function prototype declarations */

/* local module global variable declarations */

/* local inline function definitions */
/**
 * @brief This is a template function where the user can write his custom application.
 *
 * @param[in] CmdProcessorHandle Handle of the main commandprocessor
 *
 * @param[in] param2 Currently not used will be used in future
 */
void appInitSystem(void * CmdProcessorHandle, uint32_t param2);

#endif /* XDK110_XDKAPPLICATIONTEMPLATE_H_ */

/** *****/

```

8.2 Jupyter Notebook