Industrial Big Data Pipeline for Wind Turbine PHM in a Large Manufacturing Facility

Kevin Leahy[1], Colm Gallagher[2], Peter O'Donovan[3] and Dominic T.J. O'Sullivan[4]

[1,2,3,4] *School of Engineering, University College Cork, Cork, Ireland*
kevin.leahy@umail.ucc.ie; c.v.gallagher@umail.ucc.ie; peterodonovan@ucc.ie; dominic.osullivan@ucc.ie

[1,2] *Science Foundation Ireland (SFI) MaREI Centre, Environmental Research Institute, University College Cork, Cork, Ireland*

Abstract

Wind turbines generate a wealth of data which can be effectively used to improve maintenance strategies and drive down operations and maintenance (O&M) costs, which account for 20-25% of the cost of generation of wind energy. Data-driven techniques for enabling prognostic and health management (PHM) technologies have seen many successes in the space. However, managing this data, particularly in the context of an industrial facility which may have many other data streams, is a challenge. This technical brief describes the schematic of a proposed system for managing turbine data, ahead of an implementation which will see PHM techniques applied to it. The turbine in this case is attached to a manufacturing facility, so the pipeline is designed to be modular and integrate well with an existing pipeline at that facility.

1. Introduction

Operations and maintenance (O&M) of wind turbines accounts for up to 25% of the levelised cost of electricity (LCOE) generation (International Renewable Energy Agency, 2018). In order to stay competitive, these costs must be kept to a minimum. For large operators, this is usually achieved by monitoring turbines through extensive dashboards and dedicated teams. Availability contracts with the manufacturers are rigorously enforced, and any underperformance is investigated, and any corrective maintenance is carried out (Hahn, 2017). For smaller operators, e.g. manufacturing plants which make use of one or more large turbines as a source of renewable energy, the resources may not be there to dedicate this amount of effort in ensuring the turbine is running optimally. Hence, the use of artificial intelligence methods to detect any incipient faults or underperforming turbines is of added value.

Wind turbines generate a wealth of data. Efficiently ingesting, storing, processing and accessing this data is therefore very important in order to maximise its utility. By incorporating this data into a data pipeline along with the rest of a manufacturing facility's data, smart PHM apps can be easily built and deployed. This allows researchers and innovators in the space to work on building these apps, rather than cleaning and managing the data underpinning them. This brief describes how a previously developed wind turbine PHM app, described in (Leahy, Gallagher, O'Donovan, Bruton, & O'Sullivan, 2018; Leahy, Hu, et al., 2018), will be deployed using a modular data pipeline. The pipeline itself is based on earlier research in (Donovan, Leahy, Cusack, Bruton, & O'Sullivan, 2015). This is intended to give researchers and industry professionals in this space a practical insight into how theoretical models can be deployed in reality.

In section 2, a brief overview is given of the various data streams that a turbine generates. Section 3 gives an overview of the developed PHM application. In section 4, an overview of the data pipeline will be given, showing how the data is ingested and processed. This includes how the data is initially ingested, how it is processed, and how live points are sent to the trained machine learning model at the deployment stage.

2. Overview of Turbine Data

The relevant turbine data sources here come from the supervisory control and data acquisition (SCADA) system. These are: 10-minute operational data, availability data, and alarms data.

The operational data consists of various signals which are generated at 10-minute timestamps, e.g. wind speed, power, component temperatures and electrical signals. Each of these is the average value over the previous ten minute period, but can also include minimum, maximum or standard deviation. A sample of some typical turbine operational data is provided in table 1.

The availability data is also generated in 10-minute times-

Table 1. 10 Minute Operational Data

TimeStamp	Wind Speed (avg.) m/s	Wind Speed (max.) m/s	Bearing Temp (avg.) °C	Power Output (avg.) kW
09/06/2014 14:10:00	5.8	7.4	19.2	367
09/06/2014 14:20:00	5.7	7.1	19.3	378
09/06/2014 14:30:00	5.6	6.5	19.8	384
09/06/2014 14:40:00	5.8	7.5	20.2	426

tamps and contains counters displaying the number of seconds the turbine spent in each availability category over the previous 10-minute period. These categories include: generating, available but not generating due to weather or grid conditions, downtime, and time spent in repairs/maintenance.

The turbine alarm system provides similar information, but in a more granular fashion. Every time an event happens on the turbine, e.g. its status changes from generating to not generating due to low wind speed, or when the grid conditions change and turbine power output is curtailed, or when the turbine goes offline due to a system fault, an alarm message is generated. These have different severities, ranging from simply providing non-critical information to giving high severity alerts where immediate maintenance action must be taken. These messages are generated instantaneously rather than in 10-minute intervals. Typical faults range from issues like pitch motor control problems, which simply require the turbine to be reset, up to main component failure which can result in days or weeks of downtime. A sample of turbine alarm system data is provided in table 2.

3. OVERVIEW OF PHM APPLICATION

In this work, a data pipeline is described which aims to deploy a previously-developed PHM application. Classification is a proven technique for turbine fault detection and prediction, as described in (Godwin & Matthews, 2013; Leahy, Hu, et al., 2018), and the particular technique used here is described in (Leahy, Gallagher, et al., 2018).

This technical brief is intended to demonstrate how this PHM application is deployed on an existing manufacturing site. It was decided to integrate into an existing data pipeline at this site. However, had there not been an existing framework for data storage, the process described here would still be used. This is because a scalable, integrable solution allows any future turbines at this site or indeed turbines from other sites to feed into a single data repository. Data can be shared across PHM applications and the resources for these applications can be scaled up or down as needed.

3.1. Training Set

In any machine learning classification problem, the training data must be labelled. In this case, turbine alarms and availability data are used to identify periods of fault or fault-free operation on the turbine. A series of rules are used to establish times in which the turbine was operating in certain modes of operation, described in (Leahy, Gallagher, et al., 2018). These include being purposefully curtailed due to shadow or noise-curtailment restrictions, grid requirements, weather conditions, or times when the turbine was manually shut down for maintenance or planned repairs. These are in contrast to fully nominal operation, and periods when the turbine was shut down due to a fault condition. In the case of the classifiers used in this app, this consists of the following:

1. Clean the alarms and operational data to remove any incorrect or duplicate entries
2. Label the operational data for when the turbine was in various different modes of operation
3. Remove any periods for which the classifiers are not being trained, e.g. non-fault related shut-downs or curtailments
4. Label the ”pre-fault” samples, i.e. samples which occur leading up to a relevant fault-class sample within the ”pre-fault window”
5. Remove the fault samples, so that the only remaining samples are (i) pre-fault and (ii) fault-free data

This methodology is largely automated once the rules for determining which modes the turbine was operating in are established. The methodology for building this training set is described more fully in (Leahy, Gallagher, et al., 2018).

4. DATA PIPELINE

The data pipeline described in this study is intended to build upon earlier work (Donovan et al., 2015; O'Donovan, Leahy, Bruton, & O'Sullivan, 2015). These works describe a robust data pipeline for use in large-scale manufacturing facilities. Since publication, a version of the described data pipeline has been implemented in a facility, and the implementation described below builds on this in order to integrate the data from a wind turbine installed at the facility.

The pipeline is split into three parts: (i) Loading Historical and Live Data, which describes the data ingestion process, both in terms of the initial historical data upload, and how live data points are stored going forward; (ii) Building Initial Model, which describes how the historical data is used to train and store a machine learning model through which live data will be run at deployment; (iii) Deployment, which describes how live points are sent through the trained model and the results communicated with the end-user at a local level.

Table 2. Sample alarm system data from a given day

t_s	t_e	Code	Description	Category	Severity
02:13:38	07:56:14	a_{41}	Normal Operation	No Fault	Information
07:56:14	08:37:32	a_{91}	Low wind cut out	Weather	Information
08:37:32	23:44:02	a_{22}	Normal Operation	No Fault	Information

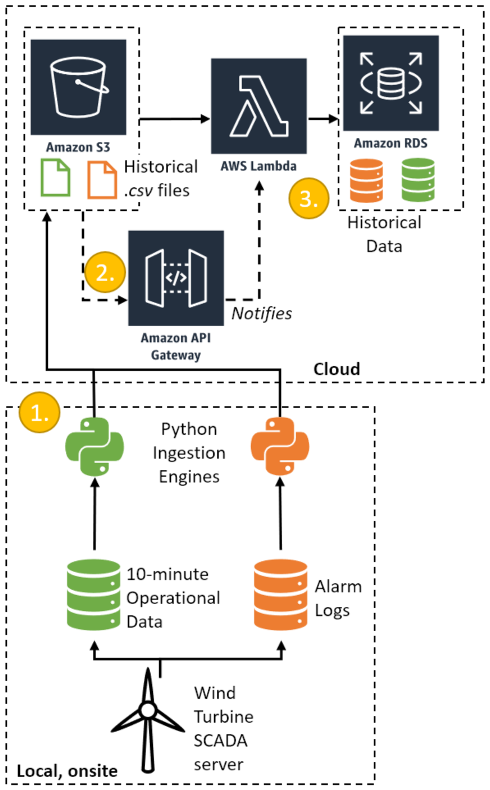

Figure 1. Historian upload process

4.1. Loading Historical and Live Data

The initial data upload process can be seen in figure 1.

As previously discussed, the wind turbine generates operational data points once every ten minutes, while the alarm system generates a new data point instantaneously when an alarm message is generated. Every time a new operational data point is generated, it is exported to a local folder in CSV format. A separate CSV is sent holding all of the alarms generated in that 10-minute period. The python ingestion engines are locally deployed scripts which monitor these local folders for new files. When the script detects a new file, it uploads it to Amazon S3, a file storage facility. This is shown in **step 1** of figure 1. This is similar to the ingestion step described in (Donovan et al., 2015).

An API gateway then communicates with AWS Lambda in **step 2**, which is an event-driven serverless computing service, typically used for light tasks and to send jobs to computing resources. In this case, it simply transfers the data to a data historian running on Amazon RDS **step 3**. These steps are similar to the queue service communicating with the "aggregate and contextualise data for analysis" through a subscription service described in (Donovan et al., 2015).

The first time this process is run, the entire local SCADA database is exported and uploaded via CSVs. From then on, the service uploads and adds to the historian every ten minutes as normal.

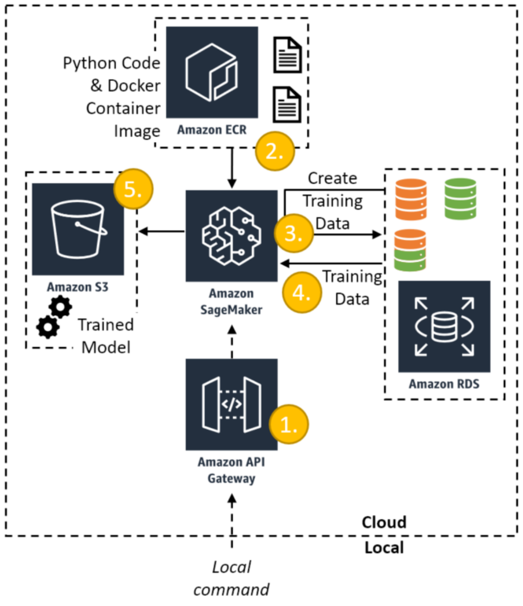

Figure 2. Data cleaning, labelling and training process

4.2. Building Initial Model

The framework for the cleaning and labelling of the training data, as well as of the actual training step for PHM apps themselves can be seen in figure 2. SageMaker is a computing resource for the construction, training and deployment of machine learning models. Amazon Eleastic Container Registry (ECR), meanwhile, is storage for the python machine learning code, and a docker container image. Docker can be thought of as a lightweight virtual machine, which can dynamically use CPU and memory from SageMaker according to the model requirements, while RDS acts as the data store.

Step 1 in the process is that a local command is sent through the API to point SageMaker to the location of the docker container image and the python code which it will run, stored

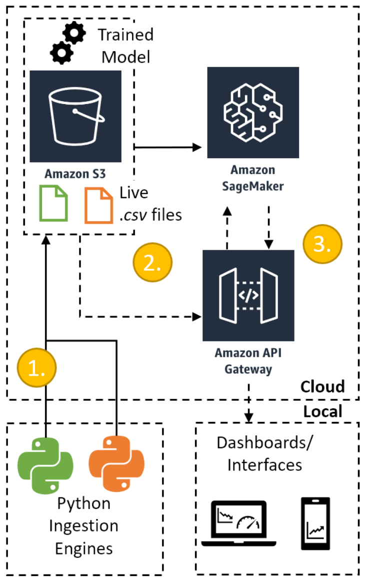

Figure 3. Deployment stage overview

on ECR. The docker container, stored on ECR, contains the script and dependencies needed to initialise and run the python code (also stored on ECR). The python code itself contains all of the steps necessary to clean, contextualise and train on the data. Docker also points SageMaker to the location of the training data in RDS. The docker container script and python code are **step 2**.

In **step 3**, the python code cleans and labels the training data according to the requirements of the particular PHM application. This is stored as a separate database in RDS in case any future updates are needed to the model, or for other PHM apps. In **step 4**, this labelled training data is fed into the python model, which again uses SageMaker computing resources to run the model. Finally, in **step 5**, the trained model is stored in S3 so that live data points can be run through it at the deployment stage.

4.3. Deployment

The deployment stage is outlined in figure 3. In **step 1**, CSV files are sent up every ten minutes, as described in section 4.1. Note that these are stored in the historian as per figure 1 (not shown here for posterity). When these are added, the API notifies SageMaker in **step 2**, which processes the CSV files to get them into the correct training format, before importing the stored model from S3 feeding the live points to it. In **step 3**, the result of this, i.e. whether or not a maintenance investigation is needed, is communicated back to the API gateway which then sends the result back locally, such as to the internal maintenance management system or off-site to the turbine maintenance provider. In this way, each step of the pipeline is a modular, customisable process that can be changed as plant requirements and maintenance strategies are updated.

5. CONCLUSION

This brief outlines a data pipeline and framework for managing a wind turbine maintenance application. The pipeline is based on cloud technologies which enable it to be modular and robust, allowing components to be easily changed and updated, or adapted for other applications. This particular use-case involves a wind turbine attached to a manufacturing plant, so is designed to integrate with existing systems for lightweight decision making. A similar version of this pipeline is currently implemented for a different piece of equipment at a manufacturing site, and this application is designed to plug into and extend that existing framework.

ACKNOWLEDGMENT

This research was funded by Science Foundation Ireland under grant 12/RC/2302 for the Centre for Marine and Renewable Energy (MaREI).

REFERENCES

Donovan, P. O., Leahy, K., Cusack, D. Ó., Bruton, K., & O'Sullivan, D. T. J. (2015). A data pipeline for PHM data-driven analytics in large-scale smart manufacturing facilities. In *Annual conference of the prognostics and health management society 2015* (Vol. 6, pp. 1–10).

Godwin, J. L., & Matthews, P. (2013). Classification and Detection of Wind Turbine Pitch Faults Through SCADA Data Analysis. *International Journal of Prognostics and Health Management*, *4*, 11.

Hahn, B. (2017). *Recommended Practice 17: WIND FARM DATA COLLECTION AND RELIABILITY ASSESSMENT FOR O&M OPTIMIZATION* (Tech. Rep. No. May). IEA Wind.

International Renewable Energy Agency. (2018). *Renewable Power Generation Costs in 2017* (Tech. Rep.).

Leahy, K., Gallagher, C., O'Donovan, P., Bruton, K., & O'Sullivan, D. (2018, jul). A Robust Prescriptive Framework and Performance Metric for Diagnosing and Predicting Wind Turbine Faults Based on SCADA and Alarms Data with Case Study. *Energies*, *11*(7), 1738. doi: 10.3390/en11071738

Leahy, K., Hu, R. L., Konstantakopoulos, I. C., Spanos, C. J., Agogino, A. M., & O'Sullivan, D. T. (2018). Diagnosing and Predicting Wind Turbine Faults from SCADA Data Using Support Vector Machines. *International Journal of Prognostics and Health Management*, *9*(1), 1–11.

O'Donovan, P., Leahy, K., Bruton, K., & O'Sullivan, D. T. J. (2015, dec). An industrial big data pipeline for data-driven analytics maintenance applications in large-scale smart manufacturing facilities. *Journal of Big Data*, *2*(1), 25. doi: 10.1186/s40537-015-0034-z