ИСПОЛЬЗОВАНИЕ МАШИННОГО ОБУЧЕНИЯ В IoT-ШЛЮЗАХ: АРХИТЕКТУРА И СЦЕНАРИИ

МАКСИМ ХЛУПНОВ

В статье демонстрируются варианты архитектуры для решения задачи по поиску аномалий в работе оборудования на основе анализа потоков его телеметрии методами машинного обучения (Machine Learning, ML). Поясняется, как можно получить и запустить ML-модели на примере задачи предсказания отказов в работе электродвигателей. Автору хотелось, чтобы статья получилась максимально практической, полезной специалистам по индустриальной автоматизации, поэтому математические формулы опущены и в конце публикации приведены ссылки на исходный код решения.

Использование моделей машинного обучения для предиктивного обслуживания промышленного оборудования на первый взгляд кажется непростой задачей. Между тем уже существует проверенный набор программного обеспечения с открытым исходным кодом и облачных сервисов, применение которых позволяет упростить и ускорить получение результата — систем предиктивного информирования операторов о появлении аномалий в работе оборудования.

ОПРЕДЕЛЕНИЕ АНОМАЛИЙ В РАБОТЕ ОБОРУДОВАНИЯ

Использование ML для определения аномалий в работе оборудования является первым шагом на пути к построению системы предиктивного обслуживания.

В классической системе удаленного мониторинга существует оператор, который смотрит на графики телеметрии оборудования (например, как на рис. 1). Оператор прошел обучение и знает, какими должны быть показатели телеметрии у нормально функционирующей системы. Такой подход работает, когда есть возможность постоянного наблюдения за удаленным процессом, однако требует постоянного внимания оператора, дает сбои, когда нужно отслеживать работы тысяч устройств, не оптимален, когда нужно улавливать тренды в изменении телеметрии на протяжении длительного времени для предупреждения отказов.

Простой контроль граничных значений телеметрии, который может быть реализован штатными средствами большинства систем удаленного мониторинга, также не всегда является решением. Так, например, у электродвигателя превышение номинального тока при пуске, как правило, не свидетельствует о какой-либо неисправности, но может приводить к ложной отправке уведомлений об отказе из-за формального превышения граничных значений. Увеличение числа условий и проверок в итоге может сделать мониторинг сложноподдерживаемым и малоуправляемым.

Использование классических методов машинного обучения для определения отклонения параметров работы от нормальных значений является перспективной тенденцией развития систем удаленного мониторинга и предиктивного обслуживания.

Достоинством данного метода является возможность применения алгоритмов, которые обучаются на данных, собранных в процессе нормальной работы двигателей. Фактиче-

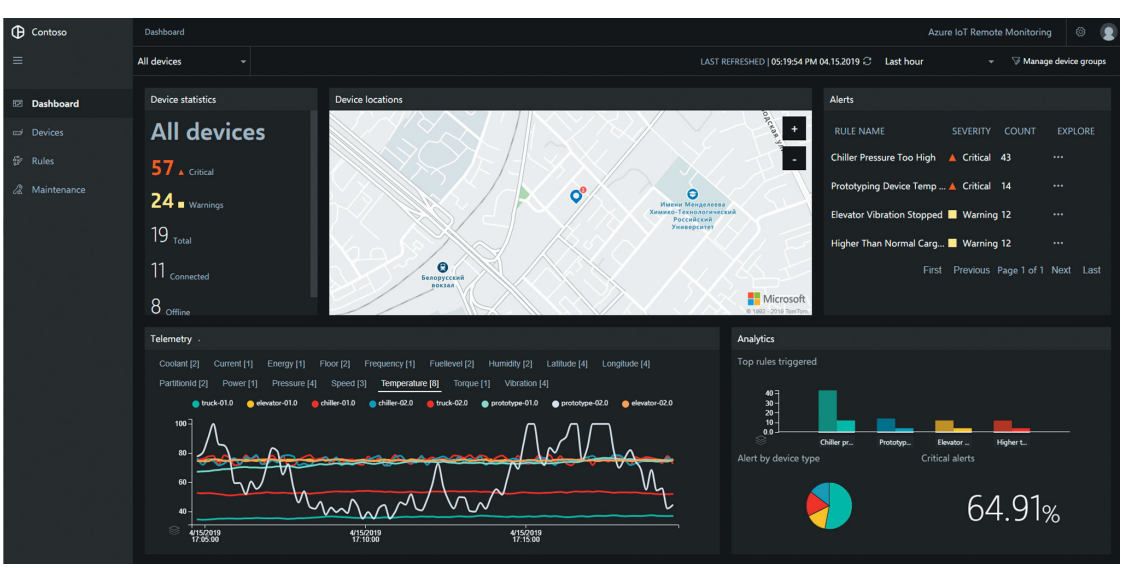

РИС. 1. ▶ Вариант реализации интерфейса оператора решения для удаленного мониторинга оборудования

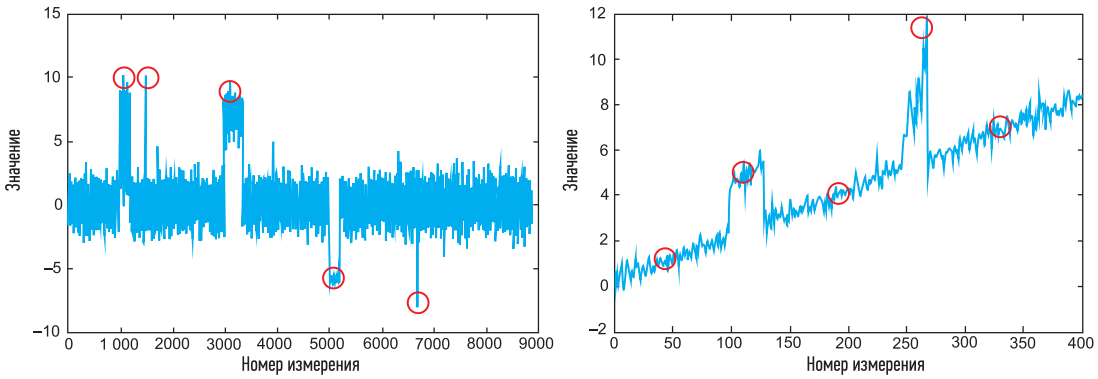

РИС. 2. ▶ Примеры аномалий. Слева: изменения уровня, выбросы и провалы. Справа: медленный восходящий тренд с изменениями

ски мы накапливаем достаточный объем статистики во время нормальной работы, а в дальнейшем применяем алгоритмы, которые ищут в структурах данных телеметрии отличия, т. е. аномалии (рис. 2). Метод основывается на предположении, что появление нетипичной телеметрии в работе установки свидетельствует о ее технической неисправности и требует внимания от обслуживающего персонала.

Использование методов машинного обучения позволяет избежать длительного и дорогостоящего создания полноценной математической модели установки, которая будет включать системы дифференциальных уравнений для описания переходных процессов в электродвигателе и прочие математические конструкции.

Это тем более актуально в связи со сменой поколений и появлением на производствах молодых сотрудников, у которых зачастую отсутствуют необходимые технические знания и опыт, что непременно сказывается на точности определения неисправностей.

Чаще всего аномалиями считаются следующие изменения в телеметрии:

- пики и провалы;
- медленные восходящие или нисходящие тренды;
- изменение уровня сигнала.

ВЫБОР АРХИТЕКТУРЫ РЕШЕНИЯ ДЛЯ ОПРЕДЕЛЕНИЯ АНОМАЛИЙ

Методов определения аномалий достаточно много. Можно долго обсуждать математику и тонкости реализации каждого метода, но критерием качества работы является проверка на реальных данных. Лучше быть сразу готовым к тому, что какой-то метод будет работать хуже и нам придется попробовать несколько вариантов перед достижением успеха. В связи с этим очень важно правильно спланировать архитектуру нашего решения — таким образом,

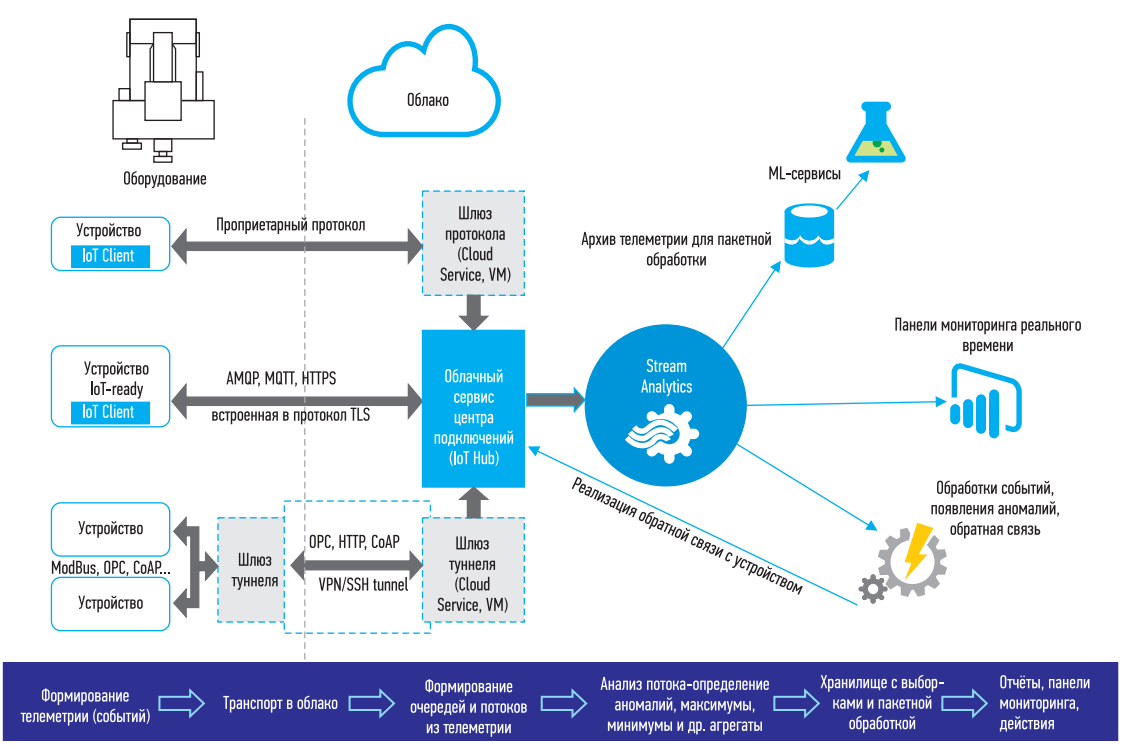

РИС. 3. ▶ Архитектура системы поиска аномалий с использованием облачной службы Azure Stream Analytics

чтобы мы могли быстро проверить модель и, если она не дает хороших результатов, перейти к следующей.

Как раз для этого нам могут пригодиться облачные сервисы, использование которых позволяет воспользоваться уже готовыми моделями и быстро получить результат. Даже если мы планируем в дальнейшем создавать собственную модель — проверить облачные сервисы и сравнить их результаты работы с собственной моделью будет крайне полезно.

Стоит отметить, что сервисы могут выполняться как в публичном облаке, так и на Edge-устройствах. При подобном подходе данные не покидают внутренней компьютерной сети, что позволяет разгрузить внешние каналы и обеспечить соответствие требованиям безопасности, которые часто запрещают передавать промышленную телеметрию оборудования за пределы компании.

Далее мы рассмотрим три варианта реализации архитектуры решения для обнаружения аномалий в работе оборудования:

- с использованием только облачных сервисов для анализа данных;
- гибридный сценарий, при котором облачные сервисы выполняются в инфраструктуре заказчика;
- полностью собственная реализация сервисов.

Облачные сервисы

Рассмотрим сначала использование исключительно облачных сервисов на примере Azure Stream Analytics, который имеет встроенную возможность определять аномалии в телеметрии (рис. 3).

В первом варианте устройства передают все данные телеметрии в облако. Определение аномалий выполняется в сервисе Stream Analytics с использованием простых SQL-подобных функций. Каждая из таких функций возвращает структуру данных, состоящую из двух полей: IsAnomaly (0 или 1), сигнализирующее о том, является ли значение телеметрии аномалией, и Score — вещественное число, отображающее, насколько сильно отклонение.

```
AnomalyDetection_SpikeAndDip(
        <scalar_expression>,
        <confidence>,
        <historySize>,
        <mode>)
OVER
([PARTITION BY<partition key>]
        LIMIT
DURATION(<unit>,<length>)
[WHEN
        boolean_expression])
AnomalyDetection_ChangePoint(
<scalar_expression>,
        <confidence>,
        <historySize>,
        <mode>)
OVER
([PARTITION BY<partition key>]
        LIMIT
DURATION(<unit>,<length>)
[WHEN
        boolean_expression])
```

Направив полученные значения IsAnomaly и Score, например, на панель мониторинга, мы можем обратить внимание оператора на отклонения в телеметрии или вызвать автоматические действия, настроив соответствующую функцию — обработчик событий.

К преимуществам архитектуры, использующей облачные сервисы для определения аномалий, можно отнести достаточную простоту реализации подобных решений и последующего внедрения. Действительно, при подобном подходе не требуется самостоятельно разрабатывать ML-модели для определения аномалий (можно воспользоваться моделями, заложенными в сервис), отсутствует необходимость создания вычислительной среды на стороне устройств, так как все данные передаются в облако и анализируются уже там. Кроме того, использование сервиса Stream Analytics позволяет анализировать поток данных от каждого устройства индивидуально, сервис будет обучаться на телеметрии каждого устройства с учетом всех его особенностей.

Однако во многих случаях необходимость применять облако для анализа телеметрии может являться существенным недостатком. Обычно возникающие проблемы связаны с плохими или дорогими каналами связи, а также с требованиями конфиденциальности. И то, и другое является препятствием для передачи телеметрии устройств в облако и, соответственно, делает невозможным использование облачного сервиса для определения аномалий.

Далее рассмотрим гибридную архитектуру решения, которая позволяет совместить преимущества применения готовых облачных сервисов с требованием производить анализ телеметрии локально, без ее передачи в облако.

IoT Edge Gateway и граничные сервисы для поиска аномалий

При гибридном подходе анализ потоков телеметрии выполняется ближе к источникам данных, на стороне устройств. Для его реализации необходимо наличие контроллера — IoT-шлюза на базе свободно распространяемой платформы с открытым исходным кодом IoT Edge, поддерживаемым Microsoft. IoT Edge позволяет переместить логику из облачных сервисов на контроллер, которым может быть любой компьютер с операционной системой Linux или Windows, поддерживающей контейнеры Moby / Docker на платформе x64/ARM32.

Кроме того, платформа реализует протокол обмена данными между модулями внутри шлюза, удаленное управление настройками и уставками модулей, а также продуманную систему безопасности на основе цифровых сертификатов и шифрования транспортного протокола.

IoT Edge состоит из трех компонентов:

- Модулей IoT Edge — физически контейнеров Moby / Docker, в которых исполняются сервисы, «приезжающие» из облака на граничное устройство и реализующие прикладную функциональность. Их создает как сам Microsoft, так и широкое комьюнити Intelligent Edge, также их может написать любой желающий на любом языке программирования. Среди модулей доступна, например, реализация промышленных протоколов Modbus, Opc-ua и т. д., расчет агрегатов телеметрии, исполнение моделей машинного обучения, анализ аномалий и т. д.
- IoT Edge runtime — сервиса операционной системы, который выполняется на каждом устройстве IoT Edge и обеспечивает обмен сообщениями, его безопасность и жизненный цикл модулей на нем.
- Интерфейса с облаком — он позволяет централизованно управлять настройками модулей и runtime из облака.

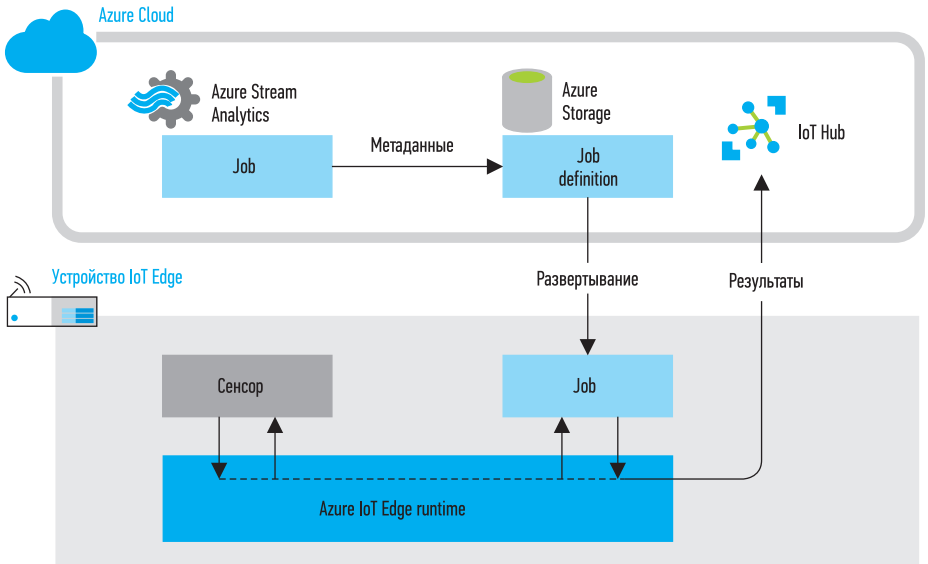

РИС. 4. Развертывание облачного сервиса Azure Stream Analytics в виде контейнера

Таким образом, мы будем по-прежнему пользоваться готовым модулем Stream Analytics, но развернем его уже в виде контейнера на Edge-устройстве (рис. 4).

В результате все наши данные будут анализироваться на Edge-устройстве, и мы будем полностью управлять тем, какую информацию передавать в облако. Например, можно передавать только уведомления о появлении аномалий и телеметрию, которая ей соответствовала.

Создание собственной модели

Уже готовые модели могут нас не устроить. Например, если мы хотим учитывать не только аномалии в изменении каждого параметра в отдельности, но и функции взаимозависимости параметров — например, тока от скорости. Если нас не устраивают модели, которые предоставляют готовые сервисы, у нас есть возможность создать свою собственную. Для создания своей модели определения аномалий обычно пользуются одной из реализаций параметрических методов или методов классификаций, поэтому ниже приведем краткую информацию о них.

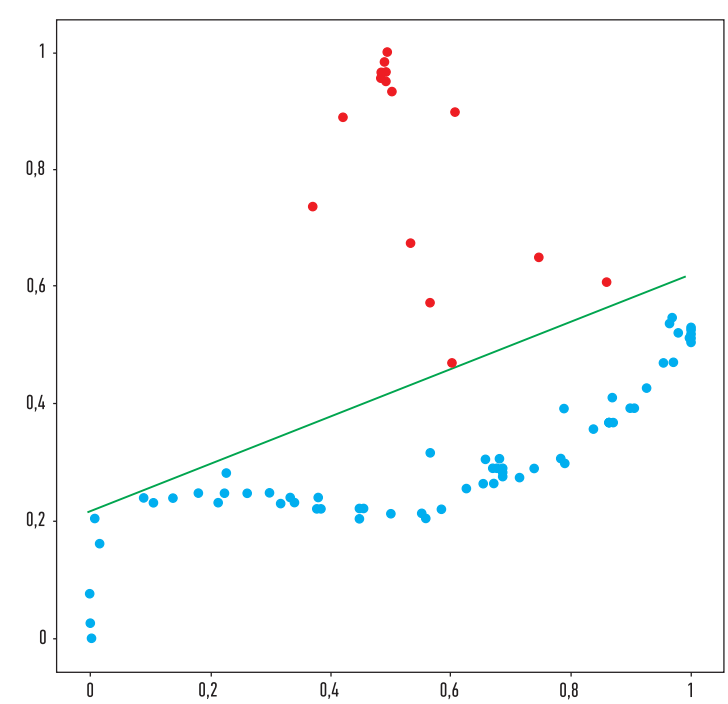

РИС. 5. Отделение аномальных значений от основной выборки на базе зависимости силы тока от скорости вращения электрического двигателя. Функция-разделитель для простоты представлена как прямая линия

Параметрические методы основаны на оценке вероятности появления подобных значений телеметрии при нормальной работе двигателя. Если вероятность ниже заданного порога, то мы можем считать данную телеметрию аномалией или выбросом в данных. Подобрать величину порога можно так, чтобы в него попадали значения телеметрии, про которые точно известно, что они являются аномальными. Алгоритмы поиска аномалий обычно относят к так называемому обучению без учителя, так как в данных телеметрии, которую мы предварительно собираем с нашей установки, информации об аномальной работе либо нет вообще, либо ее очень мало, и мы даже не знаем, какие именно данные являются аномальными.

В теории машинного обучения данный метод называется параметрическим, поскольку он основан на подборе параметров распределения, при которых значения собранной нами телеметрии (мы считаем ее нормальной) получают наибольшую вероятность.

Методы классификации предполагают отделение обучающей выборки (нормальных значений телеметрии) от аномалий. На графике это разделение можно визуализировать как кривую между нормальной телеметрией и аномальными выбросами (рис. 5). Наиболее часто используются алгоритмы логистической регрессии, деревьев решений и другие методы классификаций.

Все эти алгоритмы пришли из статистики в середине прошлого века, очень популярны и достаточно просты. Более того, они имеют проверенную реализацию на популярных языках программирования и объединены в хорошо задокументированные библиотеки.

В процессе проектирования архитектуры прикладного решения нам необходимо предусмотреть вычислительные мощности для обучения и исполнения моделей.

Обучение моделей — это разовый процесс, после его завершения вычислительные мощности не требуются — до следующей итерации, ставящей целью повысить качество обработки данных. Результатом обучения является сохраненный файл модели (обычно Python pickle-файл), содержащий исходный код программы на Python и состояние обученной модели.

Процесс обучения модели на Python для метода классификаций может быть описан такими фрагментами кода на Python:

```
# Подключение стандартных
библиотек OCC
import pickle
from sklearn import tree

# Обучение модели
        clf1 = tree.
DecisionTreeClassifier()
        clf1 = clf1.fit(Normal_
Telemetry_Data, Anomal_
Telemetry_Data)
# Сохранение состояния модели
и ее запись в pickle-файл на диск
f = open('model.pkl', 'wb')
pickle.dump(clf1, f)
f.close()
```

Процесс исполнения моделей производится постоянно для каждого сообщения с телеметрией установки на наличие аномалий. Он включает загрузку в память pickle-файла модели и проверку телеметрии — выполнение классификации. Результатом является единственное значение — флаг Anomaly (true/false).

```
# Подключение стандартных
библиотек OCC
from sklearn.externals import
joblib
from azureml.core.model import
Model
# Загрузка модели в память
model_path = Model.get_model_
path('model.pkl')
model = joblib.load(model_path)
# Классификация строки
с параметрами телеметрии
на предмет аномалий
input_json = json.loads(telemetry_
str)
pred = model.predict(input_df)
if pred[0] == 1:
input_json['anomaly']=True
else:
input_json['anomaly']=False
```

Важно помнить, что обучение моделей требует значительных вычислительных ресурсов и проводится на основе большого объема исторической телеметрии. Таким образом, для обучения моделей обычно применяется облако — как ресурс с неограниченными вычислительными возможностями, которые можно использовать, а потом отказаться от них.

Для использования модели нам понадобится выполнять указанные выше функции для каждого значения в потоке телеметрии. Чтобы этого добиться, мы можем упаковать модель в модуль IoT Edge. В результате IoT Edge runtime будет подавать значения выбранной нами телеметрии в наш модуль машинного обучения, который будет запускать модель и выполнять проверку.

ПРИМЕР РЕШЕНИЯ ДЛЯ АНАЛИЗА РАБОТЫ ЭЛЕКТРИЧЕСКИХ ДВИГАТЕЛЕЙ

Выше мы рассмотрели все составные части решения, и теперь нам осталось собрать их вместе и применить для анализа аномалий в прикладной задаче. В качестве примера такой задачи возьмем один из проектов по анализу аномалий в работе электрических двигателей, оснащенных частотными преобразователями. Искать аномалии мы будем путем анализа значений силы тока, частоты, скорости, момента и мощности электродвигателя. Решение состоит из асинхронного электродвигателя, который управляется частотным преобразователем (в нашем сценарии использовались частотные преобразователи ABB ACS 580). Частотный преобразователь, в свою очередь, опрашивается IoT-шлюзом HPE GL 20 IoT, на котором установлен Azure IoT Edge под управлением операционной системы CentOs 7.1 (рис. 6) по протоколу Modbus TCP. При промышленном использовании один IoT-шлюз может опрашивать и анализировать телеметрию нескольких десятков частотных преобразователей одновременно, например через сеть промышленного Wi-Fi Aruba.

Чтобы шлюз начал выполнять функции по определению аномалий, нам необходимо наполнить его

РИС. 6. ◄
Компонент системы на демостенде в Microsoft Technology Center

модулями (рис. 7). Исходный код модулей уже написан. Нам только нужно составить файл конфигураций и загрузить его через облачный интерфейс. После этого все модули в виде контейнеров будут загружены и запущены шлюзом. Каждый модуль содержит набор настроек, которые позволяют ему осуществлять свои функции (например, IP-адреса частотных преобразователей). Настройки также являются частью файла конфигурации.

Первый модуль — стандартный адаптер протокола ModBus от Microsoft. Он загружается в контейнере mcr.microsoft.com/azureiotedge/modbus, получает телеметрию электродвигателя через частотный преобразователь по протоколу Modbus TCP и передает ее в следующий модуль. В настройках модуля мы указываем IP-адреса частотных преобразователей, а также номера всех регистров, которые нам необходимы для расчета прикладных значений физических величин.

```
«modbus»: {
«properties.desired»: {
«PublishInterval»: «2000»,
«SlaveConfigs»: {
«acs580-01»: {
«SlaveConnection»:
«10.23.171.171»,
«HwId»: «ABB-ACS580-01-
02A6-4»,
«Operations»: {
«Speed»: {
«PollingInterval»: «1000»,
«UnitId»: «1»,
«StartAddress»: «400101»,
«Count»: «1»,
«DisplayName»: «Speed»,
«CorrelationId»: «Speed»
},
«SpeedScale»: {
«PollingInterval»: «1000»,
«UnitId»: «1»,
«StartAddress»: «404601»,
«Count»: «1»,
«DisplayName»: «SpeedScale»,
«CorrelationId»: «Speed»
},
«Frequency»: {
«PollingInterval»: «1000»,
«UnitId»: «1»,
«StartAddress»: «400106»,
«Count»: «1»,
«DisplayName»: «Frequency»,
«CorrelationId»: «Frequency»
},
«FrequencyScale»: {
«PollingInterval»: «1000»,
«UnitId»: «1»,
«StartAddress»: «404602»,
«Count»: «1»,
«DisplayName»: «FrequencyScale»,
«CorrelationId»: «Frequency»
}
...
}
}
}
```

Наш следующий модуль получает значения регистров от модуля Modbus и рассчитывает физические величины. Фактически он выполняет простые арифметические действия, которые описаны в документации частотного преобразователя. Этот модуль — custom-реализация, но он достаточно простой, а его исходный код опубликован. Настройки модуля выглядят так:

```
«abbDriveProfile»: {
«properties.desired»: {
«SignalConfigs»: {
«Speed»: {
«ValueFormula»: «Speed *
SpeedScale / 20000.0»,
«ValueType»: «Double»,
«ValueUnit»: «rpm»
},
«Frequency»: {
«ValueFormula»: «Frequency *
FrequencyScale / (10.0*20000)»,
«ValueType»: «Double»,
«ValueUnit»: «Hz»
},
...
}
}
}
```

Следующим модулем является собственно код, отвечающий за поиск аномалий. В зависимости от того, какой способ определения аномалий мы выбрали, это могут быть как настройки ASA, так и реализация на Python собственной разработки. Параметров у такого модуля будет совсем немного.

Чтобы наша информация отображалась для оператора, мы выведем

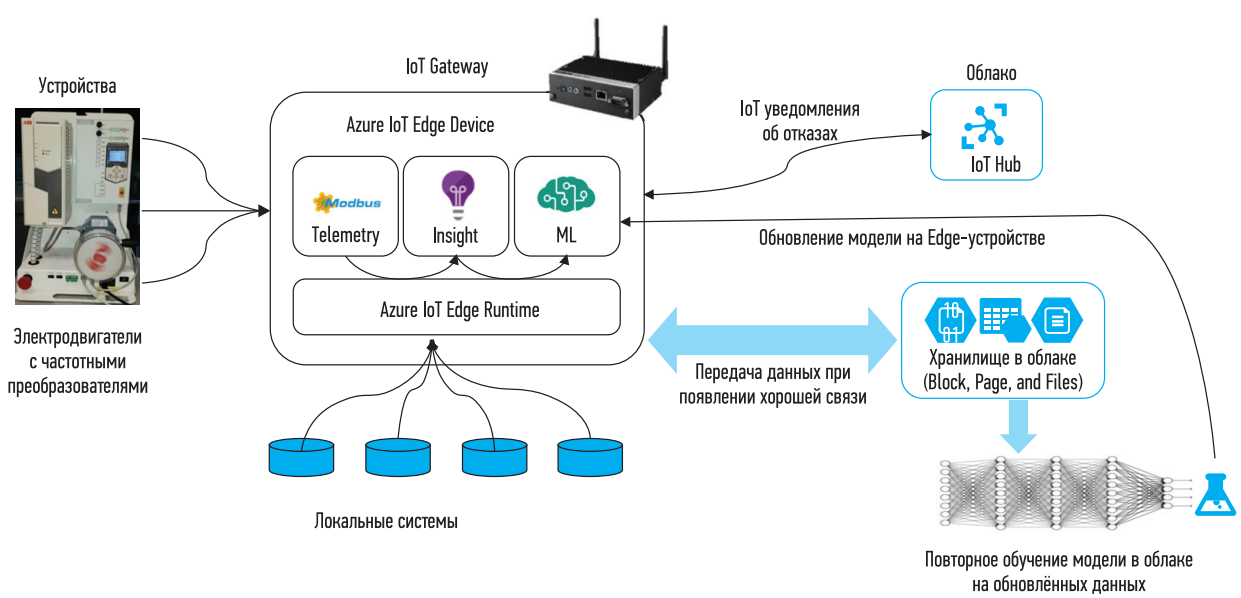

РИС. 7. Схема архитектуры системы анализа аномалий в работе электродвигателя

ее в решении с открытым исходным кодом Remote Monitoring от Microsoft. Для этого опишем поля телеметрии, которые будем передавать, в настройках модуля:

```
«DeviceTypes»:{
«Drive»: {
«TelemetryFields»: «Speed,Torque,Frequency,Current,Power,Energy»,
«ReportValueUnits»: true
}
}
```

Чтобы модули могли передавать данные и общаться между собой, в IoT Edge доступен так называемый маршрут. В маршруте связываются вход и выход модулей, а также определяется условие фильтрации выборки.

```
«modbusToAbbAcsEdgeProfile»: «FROM /messages/modules/modbus/outputs/modbusOutput INTO BrokeredEndpoint(\»/modules/abbDriveProfile/inputs/driveProfileInput\»)»,
«abbDriveProfileToRemoteMonitoringGateway»: «FROM /messages/modules/abbDriveProfile/outputs/driveProfileOutput INTO BrokeredEndpoint(\»/modules/abbRemoteMonitoringGateway/inputs/gatewayInput\»)»,
«abbRemoteMonitoringGatewayToIoTHub»: «FROM /messages/modules/abbRemoteMonitoringGateway/outputs/* INTO $upstream»,
«abbMonitoring2ML»: «FROM /messages/modules/abbRemoteMonitoringGateway/outputs/driveProfileOutput INTO BrokeredEndpoint(\»/modules/driveAnomalyDetection/inputs/AnomalyDetectIn\»)»,
«driveAnomalyDetectionToIoTHub»: «FROM /messages/modules/driveAnomalyDetection/outputs/* INTO $upstream»
```

В случае если обнаружены аномалии, оператору отправляется соответствующее уведомление.

Интересной особенностью представленной архитектуры является тот факт, что собственно промышленные данные могут никуда не передаваться, что позволяет соответствовать любым ограничениям на передачу телеметрии и реализовывать работу даже в случае потери связи со шлюзом.

ЗАКЛЮЧЕНИЕ

Применение моделей поиска аномалий в работе оборудования и других методов машинного обучения является перспективным направлением, которое позволяет в режиме квазиреального времени получать информацию о сбоях в работе оборудования или появлении негативных трендов. Существует множество реализаций методов определения аномалий, существующих как в виде облачных сервисов, так и в виде библиотек для Python и других языков программирования. Применять облачные сервисы для анализа потоков телеметрии с промышленных устройств несложно, если пользоваться решениями с открытым исходным кодом для создания шлюзов граничных вычислений, таких как Azure IoT Edge.

Подобные решения позволяют использовать свои собственные модели или загружать их из облака. Описание работающего решения можно найти по ссылкам [1, 2]. ⬣

ЛИТЕРАТУРА

1. www.github.com/MaxKhlupnov/SmartHive.AbbEdge.
2. https://youtu.be/aWYECINgQ8Q.