whitehorses

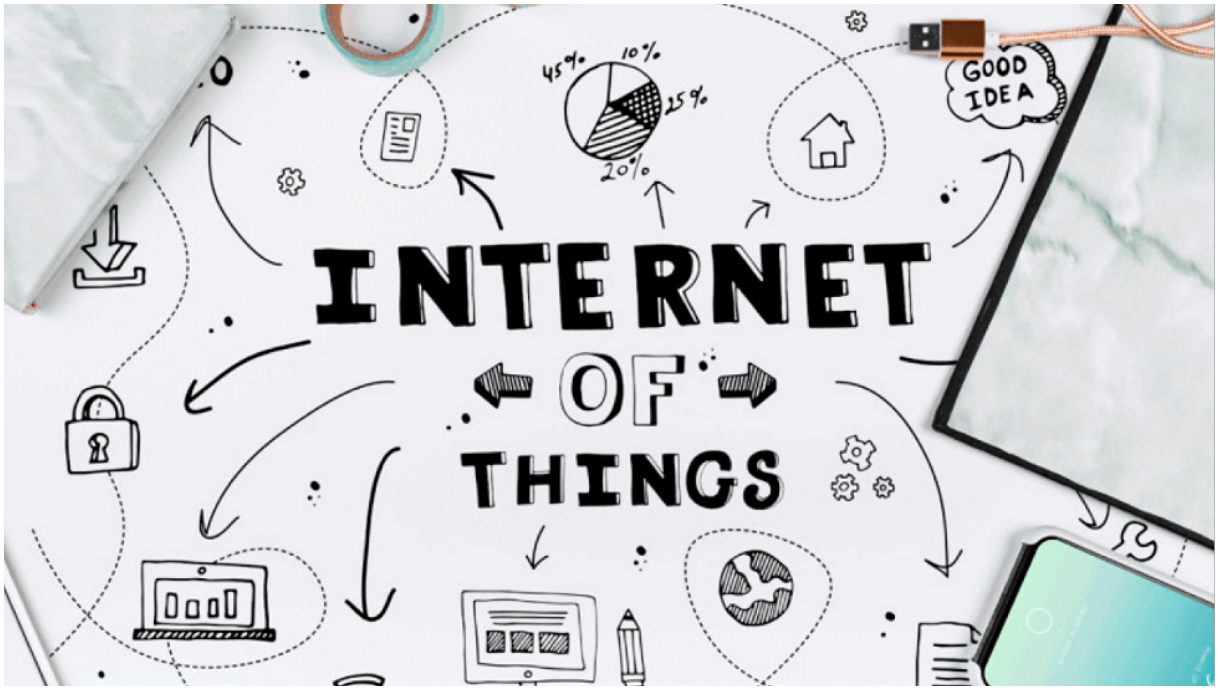
member of the human network

WHITEBOOK

Van IoT-devices naar Data Systems

April 2021

Auteur:
Peter Holtland
INTEGRATIESPECIALIST



Inleiding

Hoe komen we van normaal periodiek onderhoud van een auto tot ‘smart maintenance’?
Dit was een vraag die gesteld werd tijdens een brainstormsessie bij een leasebedrijf. Met dit Whitebook wil ik een manier beschrijven om snel een ‘proof of concept’ op te zetten voor de dataflow van sensor naar backend.

Moderne auto’s hebben een leger aan sensoren die data genereren met betrekking tot de toestand van de auto. Hoe krijgen we deze gegevens in een bruikbaar formaat op de juiste plek om er acties op uit te kunnen voeren. Met andere woorden, we willen de data zo realtime mogelijk hebben om vervolgens actief te kunnen sturen op onderhoudsmomenten voor de auto.

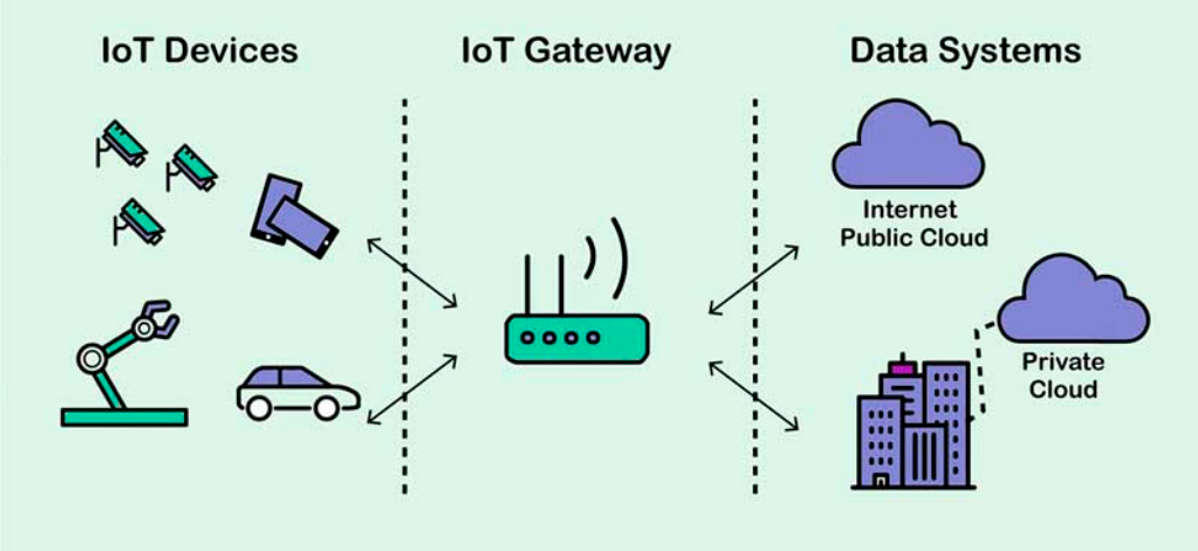
Na enig onderzoek en heel wat gegoogle kwam al snel het IoT toverwoord naar voren. Maar hoe krijgen we de data vanuit de auto op de juiste plek?
Zodra je verder kijkt bij IoT onderwerpen kom je steeds dezelfde termen tegen:

- IoT gateway,
- Message broker,
- Cloud,

om er enkele te noemen. Blijkbaar de building blocks die we nodig hebben. Hieronder nemen we deze onderdelen onder de loep en gebruiken we die vervolgens in een simpele test.

IoT gateway

Eigenlijk een soort router die zorgt dat de sensor data vanuit de auto in het juiste formaat wordt afgeleverd. Een IoT-gateway is een oplossing om IoT-communicatie mogelijk te maken, meestal device to device communicatie of device to cloud communicatie. De gateway is typisch een hardware apparaat met toepassingssoftware die essentiële taken uitvoert.
Op het meest basale niveau faciliteert de gateway de verbindingen tussen verschillende gegevensbronnen en bestemmingen.
Hier ga ik niet verder op in, zie het als de router thuis waar de laptop, mobiel en andere apparaten op aansluiten.



Message broker

We kennen veel brokers hoewel we ze wellicht niet zo noemen; JMS en Servicebus zijn brokers bijvoorbeeld. In het kort ontvangt en begrijpt de broker het bericht en weet wie het bericht moet ontvangen en zorgt daar ook voor.
Typische taken die uitgevoerd kunnen worden door een broker:

- routeerberichten naar een of meer bestemmingen;
- transformeerberichten;
- aggregatie;
- roept webservices op om gegevens op te halen.

Cloud based IoT platform

Een IoT-cloud platform is een service die IoT-devices en -toepassingen ondersteunt. Dit omvat de onderliggende infrastructuur, servers en opslag die nodig zijn voor realtime bewerkingen en verwerking.
Amazon AWS, Microsoft Azure, Google cloud bieden laagdrempelige IoT-platforms aan. Deze platforms hebben standaard services voor bijvoorbeeld Analytics maar ook voor verdere integratie naar de eigen backend zoals wellicht een database. Het IoT-platform biedt eigenlijk altijd een message broker aan voor het berichtenverkeer.

Een van de industriestandaarden die veel gebruikt wordt om het berichtenverkeer van sensor of device naar het IoT-platform te verwezenlijken is het MQTT-protocol, hier zoomen we wat verder op in.

MQTT-protocol

Message Queue Telemetry Transport (MQTT) is een lichtgewicht messaging-protocol gebaseerd op het publish/subscribe model. MQTT gebruikt een clientserver architectuur waarbij de client (zoals een sensor of ander device op een auto) een verbinding maakt met een MQTT-server (een broker genoemd) en publiceert berichten op een server topic. De broker stuurt de berichten door naar subscribers die geregistreerd zijn op het topic. MQTT is zeer geschikt voor devices en omgevingen met beperkte resources zoals processorcapaciteit, geheugen en bandbreedte.

Even verder kijken naar MQTT

MQTT is een extreem lichtgewicht message-protocol. De publish/subscribe architectuur is ontworpen om open en gemakkelijk te implementeren.
Een enkele MQTT-server kan tot duizenden externe clients ondersteunen. Deze kenmerken maken MQTT ideaal voor gebruik in een omgeving met beperkte resources, typisch het geval bij IoT-devices. Het MQTT-protocol is publish/subcribe gebaseerd, berichten worden gepubliceerd op van tevoren gedefinieerde topics op een MQTT-server of wel de message broker. Subscribers krijgen van dezelfde message broker de berichten die voor hen zijn bestemd.

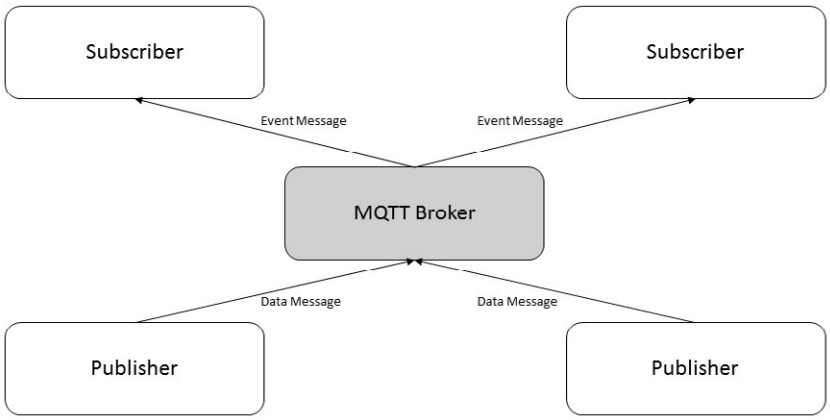


Fig1: Het schema voor MQTT is typish publish/subscribe

Het bovenstaande schema is niet anders dan gebruikelijk bij de message systemen die we gebruiken.
Het belangrijkste voordeel van MQTT zit hem in de enorm kleine footprint van het protocol, die maakt het ideaal voor gebruik bij kleine devices zoals bijvoorbeeld sensoren in een auto.
Leuk feitje: Facebook gebruik het MQTT-protocol voor de messenger-app om batterijverbruik te verminderen.
Enkele belangrijke kenmerken van MQTT:

- Gebruikt TCP / IP voor netwerkconnectiviteit
- Werkt met TLS 1.2
- MQTT biedt standaard QoS voor het afleveren van berichten onder andere ‘Exactly Once’
- Zeer eenvoudige specificatie en API’s
- De belangrijkste API’s zijn CONNECT, PUBLISH, SUBSCRIBE, DISCONNECT
- De berichten header is kort in MQTT en de kleinste pakketgrootte is 2 bytes, waardoor het ideaal is voor kleine apparaten

Data Flow

- Devices verzenden gegevens in een vooraf gedefinieerd formaat naar IoT-gateways
- Wifi, 4G/5G Bluetooth communicatie van IoT-gateway naar IoT-platform
- IoT-gateway verzendt gegevens naar topic in de message broker op IoT-platform
- Op IoT-platform wordt bericht verwerkt en afgeleverd aan subscribers

Let wel, dit is de kale dataflow zonder enige toegevoegde waarde. De toepassing krijgt echte toegevoegde waarde zodra er verder downstream iets mee gebeurt. Deze berichtenstroom kan bijvoorbeeld gekoppeld worden aan een analytische toepassing die kijkt of bepaalde waardes worden bereikt om vervolgens een taak op te starten of een workflow. Een belangrijk onderdeel van de auto overschrijdt een kritieke waarde, een rest service bestelt het onderdeel, een andere rest service boekt een afspraak bij het onderhoudscentrum. Zomaar een voorbeeld.

Voorbeeld ter illustratie

Ik heb alle onderdelen die hierboven zijn besproken gebruikt om een simpele proof of concept te maken. De taak van het genereren van data wordt gedaan door de laptop, deze simuleert als het ware een sensor. Op de laptop worden met behulp van Python en de Pika client data en berichten aangemaakt. Deze gaan via mijn huis - tuin en keuken router, de IoT-gateway, naar de message broker. Het IoT-platform in de cloud wordt verzorgd door het AWS IoT-platform.

De laptop is het device, Python/Pika is de interface op het device, mijn router de gateway en AWS IoT-platform heeft de ontvangende message broker.

Hieronder zijn korte codefragmenten voor het opzetten van de verbinding en het versturen van een bericht. Een topic met de naam ‘mijnAuto’ is aangemaakt in de message broker op het IoT-platform:

Voor het tot stand brengen van de verbinding met de broker.

```
#!/usr/bin/env python
import pika

connection = pika.BlockingConnection(pika.ConnectionParamete
rs(‹mijnAWSplatform:5672›))
channel = connection.channel()
```

Versturen bericht naar het juiste topic

```
channel.basic_publish(exchange='',
                      routing_key='mijnAuto',
                      body='autoData')
connection.close()
```

Codefragment voor het ontvangen van test berichten ter illustratie:

```
#!/usr/bin/env python
import pika, sys, os

def main():
    connection = pika.BlockingConnection(pika.ConnectionPara
meters('mijnAWSplatform:5672'))
    channel = connection.channel()

        def callback(ch, method, properties, body):
        print(" [x] Received %r" % body)

    channel.basic_consume(queue='mijnAuto', on_message_
callback=callback, auto_ack=True)

    print(' [*] Waiting for messages. To exit press CTRL+C')
    channel.start_consuming()

if __name__ == '__main__':
    try:
        main()
    except KeyboardInterrupt:
        print('Interrupted')
        try:
            sys.exit(0)
        except SystemExit:
            os._exit(0)
```

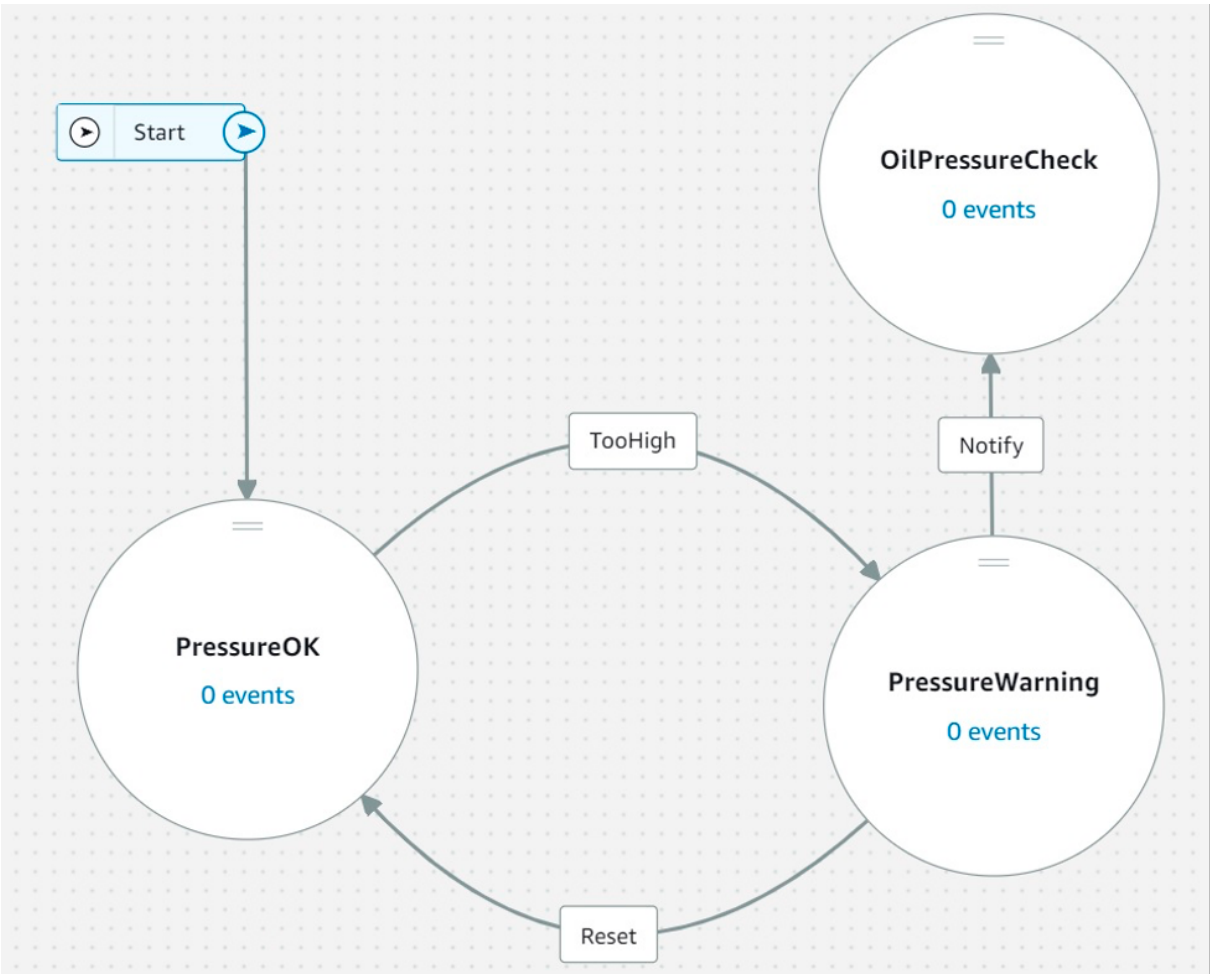


Fig 3: EventDetector service design

Op het AWS-platform staat nu een EventDetector service die de binnengekomen waarden controleert en vervolgens de status weergeeft van de desbetreffende sensor. Afhankelijk van de waarde die binnenkomt wordt in de EventDetector de status gezet. We kunnen vervolgens gaan zien op het topic wat er gebeurt zodra de sensor berichten begint te sturen.

Detectors (4)

Key value	Current state
100005	PressureWarning
100007	OilPressureCheck
100008	OilPressureCheck
100009	PressureOK

Fig 4: Huidige status van de sensoren na versturen testberichten

Elke SensorID heeft zijn eigen status, de status wordt aangepast zodra er nieuwe waarden binnenkomen. Het is niet lastig om vervolgens het model uit te breiden met bijvoorbeeld een SMS, WhatsApp- of Email-notificatie wanneer de binnengekomen waarden directe actie vereisen.

Tot slot

Met behulp van de beschreven componenten kan op een eenvoudige manier een begin gemaakt worden met het gebruiken van data uit verschillende bronnen. Met behulp van een Cloud platform kan eenvoudig data van IoT devices binnengehaald en geanalyseerd worden. Afhankelijk van signaalwaarden kunnen er verdere acties ondernomen worden. Klein beginnen en vervolgens uitbreiden.