THOMAS MORE

Realisatieplan Stage

Mathieu Du Jardin

2023 - 2024

1. Woord Vooraf

Dit stageverslag beschrijft mijn ervaringen en inzichten die ik heb opgedaan tijdens mijn stage bij Van Genechten Packaging, Imas NV. Het doel van mijn stage was om een predictive maintenance systeem te ontwikkelen (basis te leggen en mogelijkheden testen) dat gebruik maakt van trillingssensoren (vibratiesensoren) om potentiële problemen met machines te detecteren voordat ze zich voordoen. Door middel van dit systeem kunnen we de betrouwbaarheid van de machines verhogen, onderhoudskosten verlagen en onverwachte stilstandtijden minimaliseren.

In dit verslag wordt de volledige ontwikkeling en implementatie van het voorspellende onderhoudssysteem gedocumenteerd.

Tijdens mijn stage heb ik niet alleen technische vaardigheden ontwikkeld, maar ook waardevolle ervaring opgedaan in projectplanning, samenwerking en documentatie. Ik wil graag mijn dank uitspreken aan mijn stagebegeleiders en collega's bij Imas NV voor hun begeleiding en ondersteuning gedurende deze leerzame periode.

Speciale dank gaat uit naar de mensen van SICK voor hun expertise en uitleg over de trillingssensoren, en naar mijn opleiding voor de kans om deze stage te voltooien.

2. Inhoud

3. Lijst met Figuren

4. Inleiding

Tijdens mijn studie Internet of Things (IoT) heb ik de kans gekregen om een stage te lopen bij Van Genechten Packaging, specifiek bij hun dochterbedrijf Imas NV. Deze stage bood een ideale gelegenheid om zowel mijn theoretische kennis als praktische vaardigheden in IoT toe te passen en verder te ontwikkelen. De focus van mijn stage lag op predictive maintenance, een innovatief en groeiend gebied binnen IoT dat gebruikmaakt van geavanceerde technologieën om onderhoudsbehoeften te voorspellen en zo de operationele efficiëntie te verbeteren.

Gedurende mijn stage heb ik gewerkt met diverse hardware en software tools die cruciaal zijn voor IoT-ontwikkelingen.

Mijn tijd bij Imas NV was bijzonder waardevol en leerzaam, en ik ben dankbaar voor de mogelijkheid om mijn kennis en vaardigheden in een praktijkgerichte setting te verdiepen. In dit verslag zal ik de verschillende aspecten van mijn stage belichten, van de doelstellingen en de gebruikte technologieën tot de uitdagingen die ik ben tegengekomen en de resultaten die ik heb behaald.

5. Stagebedrijf

Ik heb stagegelopen bij Imas NV, een dochterbedrijf van Van Genechten Packaging.

Van Genechten Packaging specialiseert zich in het maken van hoogwaardige vouwkartonnen verpakkingen en Imas NV is de IT-afdeling binnen Van Genechten Packaging.

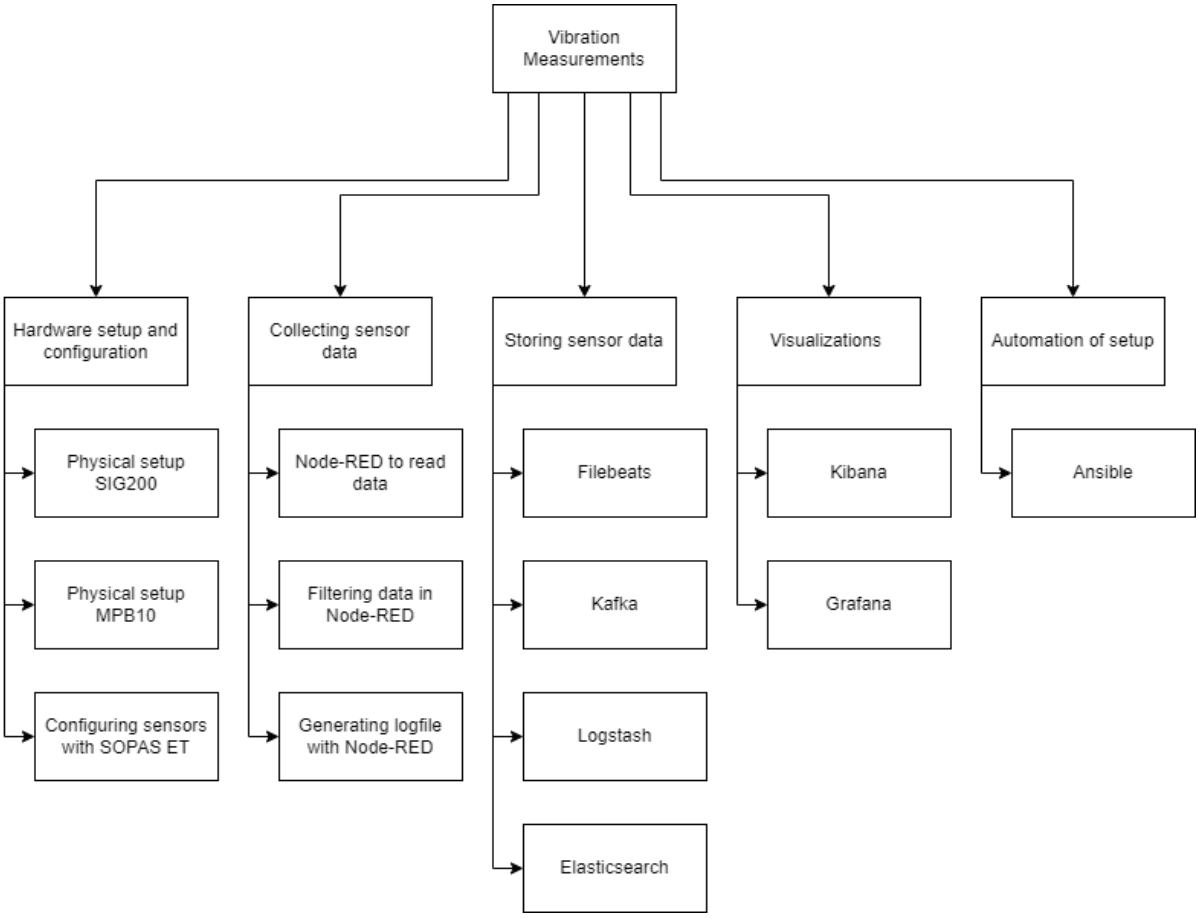
Figuur 1: Logo VGP

6. Doelstelling en Scope + WBS

Van Genechten Packaging wil kunnen voorspellen wanneer een machine problemen zal ondervinden.
Ze streven ernaar dit te bereiken door trillingssensoren op de machines te gebruiken. De trillingen zouden consistent moeten zijn wanneer alles goed functioneert, maar wanneer een onderdeel van de machine begint te falen, zouden de trillingen anders zijn. De trillingssensor is een SICK-sensor.

Het project wordt ontwikkeld op één machine, maar de oplossing moet toepasbaar zijn op meerdere machines. Dit moet automatisch gebeuren met Ansible. De gegevens van de trillingssensor worden opgeslagen in een Elastic database. Node-RED wordt gebruikt om de trillingsgegevens van de sensor te lezen. Node-RED bevindt zich op een reguliere smartbox.
Alles moet gedocumenteerd worden.

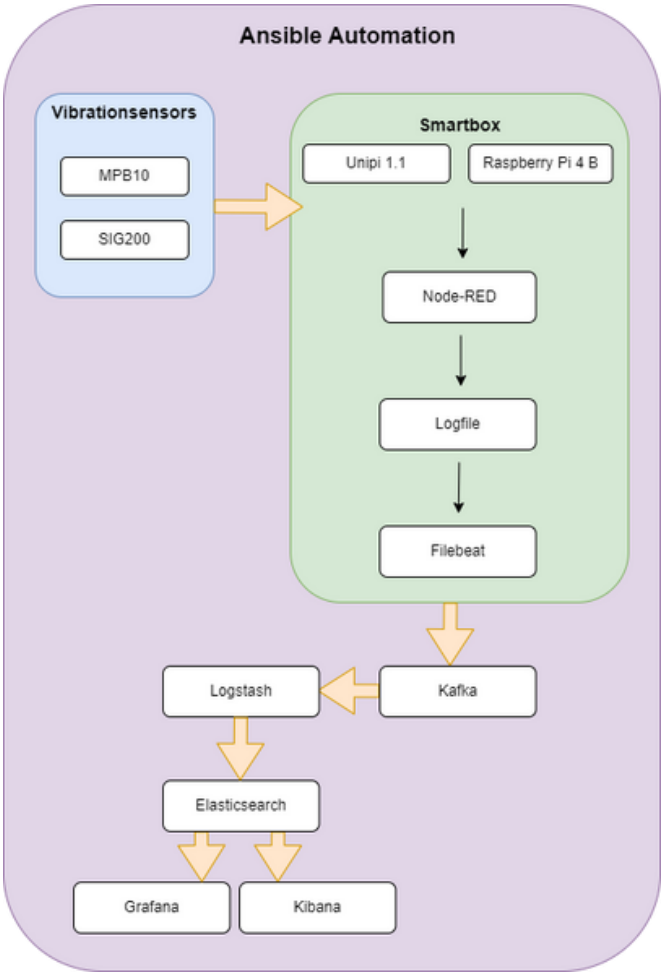
De trillingssensoren moeten correct worden ingesteld. Ze worden gelezen met behulp van Node-RED. Node-RED maakt een logbestand van de gegevens, en Filebeat stuurt deze gegevens naar Kafka. Met Logstash worden de gegevens van Kafka naar Elasticsearch gestuurd en opgeslagen. De gegevens worden vervolgens gevisualiseerd met Grafana. Dit hele proces moet geautomatiseerd worden met Ansible.



Figuur 2: Work Breakdown Structure Project

7. Overview

Hier is een algemene overview over het project:



Figuur 3: Solution Architecture Project

In de blauwe box hebben we twee trillingssensoren: MPB10 en SIG200. De MPB10 gebruikt IO-Link voor communicatie. Omdat de Raspberry Pi geen native ondersteuning heeft voor IO-Link, maken we gebruik van de SIG200, die fungeert als een IO-Link Master en toegankelijk is via een REST API.

Bij Imas werken ze met smartboxen. Deze smartboxen hebben een aantal functionaliteiten. De functionaliteit die ik moet toevoegen is: vibratiemetingen zodat ze predictive maintenance kunnen doen.

De groene doos vertegenwoordigt een smartbox die zowel een Unipi als een Raspberry Pi bevat. Op de Raspberry Pi hebben we Node-RED geïnstalleerd met een "Node-RED flow" die draait op poort 1600. Binnen deze flow gebruiken we de REST API om data uit de SIG200 te halen. Vervolgens genereren we met Node-RED een logbestand met deze trillingsgegevens, die ook in de smartbox wordt opgeslagen.

Daarna wordt de data via Filebeat naar Kafka verzonden onder het onderwerp smartboxlogging, getagd als vibrationdata.

Daarna haalt Logstash de data van Kafka op en stuurt deze naar Elasticsearch onder de index vibrationdata.

In Grafana hebben we deze Elasticsearch-index ingesteld als de primaire gegevensbron zodat we visualisaties kunnen maken.

Het hele proces wordt geautomatiseerd door Ansible.

7.1 Sensors (Hardware)

- MPB10 Multi Physics Box
  De MPB10 is een trillingssensor van SICK die communiceert via IO-Link. Omdat de Raspberry Pi en UniPi (in de smartbox) geen IO-Link-communicatie ondersteunen, hebben we de IO-Link master (SIG200) nodig.

Figuur 4: MPB10

- SIG200 IO-Link Master
  De SIG200 Sensor Integration Gateway is een IO-Link master voor het integreren van IO-Link apparaten in PLC-omgevingen en systemen op bedrijfsniveau. Het heeft een ingebouwde REST API waarmee eenvoudig gecommuniceerd kan worden.


Figuur 5: SIG200

- Config Cable
  Deze dient om de SIG200 te configureren met de Sopas ET Software.

Figuur 6: Config Cable

- Power Cable
  Deze geeft stroom aan de SIG200.

Figuur 7: Power Cable

- Ethernet Cable
  Om de SIG200 op het netwerk te krijgen gebruiken we Ethernet.

Figuur 8: Ethernet Cable

- Extension Cable (10m)
  The MPB10 komt met een ingebouwde korte kabel (paar cm). Voor langere afstanden is de extension kabel nodig.

Figuur 9: Extension Cable

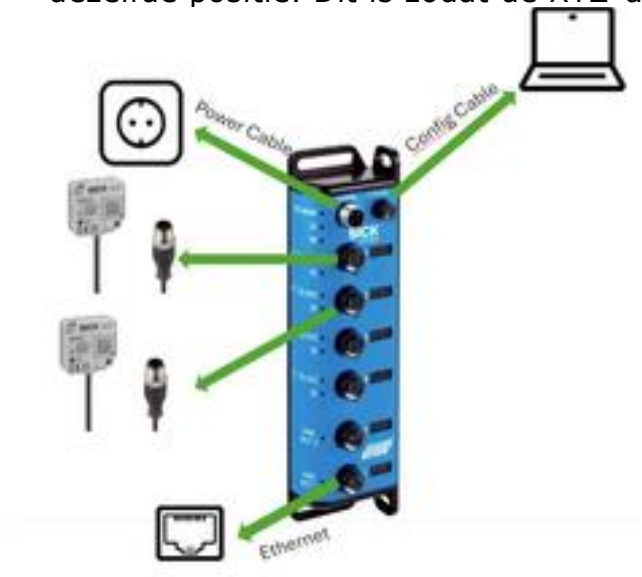
7.2 Sopas ET Software (Software)

Figuur 10: Sopas ET Logo

We gebruiken de SOPAS ET Software (Sopas Engineering Tool) om de SIG200 te configureren. In SOPAS ET is het mogelijk om allerlei informatie over de aangesloten sensoren op te halen: status van de poorten, IP-adres, data van de sensoren, enzovoort. De tool wordt ook gebruikt om de sensoren en SIG200 te configureren met IODD-bestanden, IP-adres, lees-/schrijfrestricties, enzovoort (zie later: configuratie van trillingssensor).

7.3 Fysieke Opstelling van de SIG200 + MPB10

1. Sluit de MPB10-sensoren aan op poort S1-S4 van de SIG200 (begin altijd bij poort S1, dan S2, enzovoort. Als je slechts 2 sensoren hebt, gebruik dan S1 en S2).
2. Sluit de voedingskabel aan op de POWER-poort.
3. Sluit de configuratiekabel aan op de CONFIG-poort en steek het USB-uiteinde in je pc.
4. Sluit de ethernetkabel aan op de P1-poort.
5. Plaats de MPB10-sensoren op de machine, allemaal in dezelfde positie. Dit is zodat de XYZ-as overal hetzelfde is.



Figuur 11: Verbindingen SIG200

7.4 Configuratie van de SIG200 + MPB10

Zodra de sensor fysiek is geïnstalleerd, gebruiken we de Sopas Engineering Tool om de SIG200 en MPB10 te configureren. Voor het configureren van de sensoren raden we aan de handleiding van de SIG200 te gebruiken. In de handleiding staat bij punt 7.3 uitgelegd hoe je SOPAS ET gebruikt om de sensor te configureren.

Hoe je dit doet:

1. Doe eerst de fysieke installatie.
2. Installeer SOPAS ET en start het programma.
3. Installeer de drivers (de drivers staan al op de SIG200 vanuit de fabriek, dus selecteer de optie voor de "lokale drivers").
4. Log in als maintenance (gebruiker: maintenance, wachtwoord: main) en druk op "online".
5. Dubbelklik op de SIG200.
6. Er opent een nieuw venster waar je de SIG200 kunt configureren.
7. Verander het IP-adres (zorg ervoor dat het op hetzelfde netwerk is, test met het ping-commando, het moet kunnen pingen).
8. Installeer IODD-bestanden voor de MPB10, te vinden bij IODDFinder.

Opmerking: Als een configuratie niet wordt opgeslagen, zoals bijvoorbeeld het IP-adres, zet dan de SIG200 even aan en uit.

8. Data Parameters

We gaan predictive maintenance uitvoeren op basis van trillingen, dus we richten ons uitsluitend op het meten van de trillingen. Er zijn echter verschillende soorten trillingen en diverse methoden om deze te meten. De MPB10 kan zowel a-RMS als v-RMS meten. We moeten beslissen welke van deze we gaan gebruiken voor onze metingen.

8.1 a-RMS vs v-RMS

a-RMS (acceleration RMS):
De volgende mechanische fouten hebben een sterke invloed op de a-RMS:

- Wrijving: de weerstand die ontstaat wanneer een oppervlak over een ander beweegt. Dit kan trillingen veroorzaken.
- Aanraking van machineonderdelen: wanneer 2 of meer machineonderdelen met elkaar in contact komen.
- Onbalans: in roterende componenten.
- Slijtage: wanneer een onderdeel slijt, kan dit trillingen veroorzaken door veranderingen in hun geometrie of oppervlakte-eigenschappen.
- Complicaties met smeermiddelen: onjuiste smering, verslechterende smering...

a-RMS meet hoe snel de snelheid van een object in de loop van de tijd verandert. Het wordt gebruikt om te detecteren hoe snel iets versnelt of vertraagt.

v-RMS (velocity RMS):
De volgende mechanische fouten hebben een sterke invloed op de v-RMS:

- Onbalans: in roterende componenten.
- Verkeerde uitlijning: onjuiste uitlijning van machineonderdelen.
- Relaxation: het geleidelijke settelen of aanpassen van componenten in de loop van de tijd.

v-RMS meet de snelheid waarmee een object trilt of beweegt in de loop van de tijd. Het is nuttig voor het beoordelen van de algehele trillingsniveaus en de status van de machine.

<u>a-RMS en v-RMS uitgelegd met een voorbeeld (schommel):</u>
-Acceleratie RMS (a-RMS): Denk aan acceleratie als hoe snel je snelheid verandert terwijl je zwaait. Wanneer je begint te zwaaien, duwt iemand je van achteren, waardoor je snelheid snel verandert. Dit is acceleratie. Dus, wanneer je net begint te zwaaien en de snelheid plotseling verandert, meet acceleratie RMS hoe snel die verandering plaatsvindt. Een hoge acceleratie

RMS zou betekenen dat je heel snel begint te zwaaien, wat kan gebeuren als iemand je een krachtige duw geeft.

-Snelheid RMS (v-RMS): Denk aan snelheid als hoe snel je heen en weer beweegt terwijl je zwaait. Terwijl je zwaait, beweeg je heen en weer met een constante snelheid. Dus, wanneer je al een tijdje aan het zwaaien bent en een constante snelheid hebt bereikt, meet snelheid RMS hoe snel je heen en weer beweegt gedurende een bepaalde periode. Een hoge snelheid RMS zou betekenen dat je snel heen en weer beweegt tijdens het zwaaien.

8.2 Relaxation (v-RMS)

Relaxation is het fenomeen waarbij materialen, componenten of structuren in de loop van de tijd hun vorm, positie of spanning veranderen als reactie op verschillende omgevingsfactoren of belastingen.

Hier zijn enkele voorbeelden:

- Materiaalkruip: Dit treedt op bij materialen onder langdurige mechanische belasting, zoals bij compressie, trekkracht of herhaalde trillingen. Bijvoorbeeld, een metalen component die continu aan een belasting wordt blootgesteld, kan langzaam vervormen of "kruipen" onder invloed van de aangebrachte krachten. Dit kan leiden tot veranderingen in afmetingen, posities of spanningen van de componenten, wat op zijn beurt kan leiden tot ongewenste trillingen en prestatievermindering.
- Verzakken van funderingen: Bij industriële machines kunnen funderingen of ondersteuningen in de loop van de tijd ook verzakken of verschuiven als gevolg van bodemzetting, structurele veranderingen of thermische effecten. Dit kan leiden tot ongewenste veranderingen in de uitlijning van machineonderdelen, die op hun beurt trillingen en operationele problemen kunnen veroorzaken.
- Thermische uitzetting en krimp: Materialen kunnen uitzetten of krimpen wanneer ze worden blootgesteld aan variërende temperaturen. Dit kan leiden tot microscopische verschuivingen of vervormingen van onderdelen, die zich in de loop van de tijd kunnen ophopen en trillingsproblemen kunnen veroorzaken.
- Structurele ontspanning: Sommige materialen kunnen structurele ontspanning ondergaan, waarbij interne spanningen of defecten in het materiaal geleidelijk worden geëlimineerd na fabricage of bewerking. Deze veranderingen kunnen in de loop van de tijd van invloed zijn op de mechanische eigenschappen van de materialen en kunnen van invloed zijn op de prestaties en betrouwbaarheid van machines.

8.3 Welke is het beste voor onze oplossing?

Als de machine plotseling sneller begint te draaien, resulterend in een snellere verandering in snelheid, zou het zinvol zijn om a-RMS te meten. Dit komt doordat a-RMS snelheidsveranderingen meet, wat het geval zou zijn wanneer de machine plotseling versnelt. Het helpt om een beter begrip te krijgen van de impact van die plotselinge snelheidsveranderingen op de trillingen van de machine. Denk ook aan de mechanische fouten die hierboven zijn genoemd en een sterke invloed hebben op a-RMS.

Aan de andere kant, als de machine al op constante snelheid draait en alleen maar trilt, zou het zinvol zijn om v-RMS te meten. Dit komt doordat v-RMS de constante snelheid meet en de bijbehorende trillingen wanneer de machine al draait op zijn maximale snelheid. Denk ook aan de mechanische fouten die hierboven zijn genoemd en een sterke invloed hebben op v-RMS.

Het meten van a-RMS of v-RMS hangt af van het type machine. Als je een machine hebt die gesmeerd is en je wilt meten, dan zou a-RMS nuttig zijn. Over het algemeen is v-RMS een goede optie. Voornamelijk omdat we willen meten wanneer de machine draait en trilt bij een constante snelheid. Relaxation is ook een belangrijke factor, die wordt gemeten door de v-RMS.

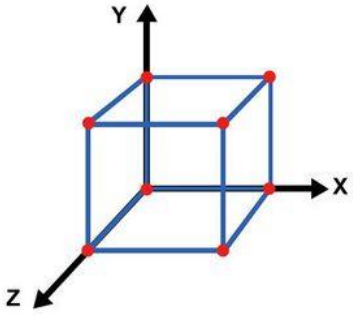
8.4 XYZ Axis

De MPB10 meet in de XYZ Axis:

- X-as: De X-as vertegenwoordigt horizontale beweging of oriëntatie. In de context van trillingsmeting geven trillingen langs de X-as bewegingen in de horizontale richting aan. Dit kunnen zijwaartse of heen-en-weer bewegingen zijn, afhankelijk van de oriëntatie van de sensor.

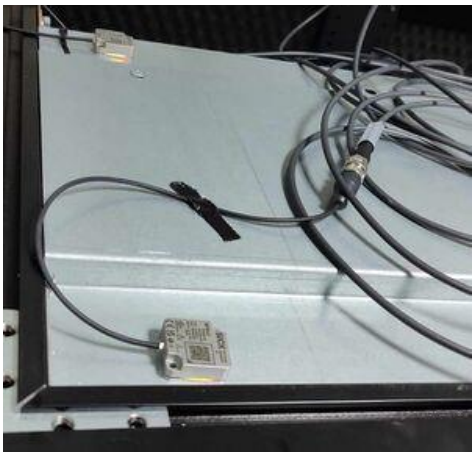
- Y-as: De Y-as vertegenwoordigt verticale beweging of oriëntatie. Trillingen langs de Y-as geven bewegingen in de verticale richting aan. Dit kunnen op-en-neer bewegingen zijn, opnieuw afhankelijk van de oriëntatie van de sensor.

- Z-as: De Z-as vertegenwoordigt diepte- of voor-achterwaartse beweging of oriëntatie. Trillingen langs de Z-as geven bewegingen in de diepte- of voor-achterwaartse richting aan. Dit kunnen vooruit-en-achteruit bewegingen zijn.



Figuur 12: XYZ Axis Figuur

Het is belangrijk dat bij het doen van de fysieke installatie, je de sensoren allemaal in dezelfde richting plaatst, zodat de XYZ-as hetzelfde is voor elke sensor.



Figuur 13: Foto Airco

Hier zie je de 2 sensoren die in dezelfde richting wijzen (kabel naar links gericht). Als je meer sensoren hebt, moet je ze ook in deze positie plaatsen (met de kabel naar links gericht).

9. Node-RED

Figuur 14: Logo Node-RED

Node-RED is een visuele programmeeromgeving die wordt gebruikt voor het creëren van IoT-toepassingen en workflows. Het stelt gebruikers in staat om snel en eenvoudig verbindingen te maken tussen verschillende apparaten en services door middel van drag-and-drop van vooraf gebouwde functieblokken, die "nodes" worden genoemd. Met Node-RED kunnen gebruikers gegevens verzamelen, verwerken, transformeren en beheren, en vervolgens acties uitvoeren op basis van deze gegevens. Het is een krachtige tool voor het ontwikkelen van IoT-toepassingen en automatiseringsscenario's zonder dat er diepgaande programmeerkennis vereist is.

9.1 Environment Variables

type	name	variable name	default value	explained
	Vibration	VBM_SIG_IP		this is the IP address of the SIG200 (the one you gave in SOPAS ET), this IP is used in the link of the HTTP request node. This way you don't have to change it in the Node-RED but you can do it with ansible.
	Vibration	VBM_AMOUNT_PORTS	4	a number from 1-4. It depends on how many MPB10s you want to connect to your SIG200.
	Vibration	VBM_LOG_FILE		this is the logfile path where the vibration data is stored

Figuur 15: Environment Variables Tabel

Environment variables in Node-RED zijn variabelen die worden gebruikt om configuratie-instellingen en gevoelige informatie op te slaan die door de Node-RED-runtime wordt gebruikt.

Bij Imas NV worden de environment variabelen geconfigureerd in Ansible.

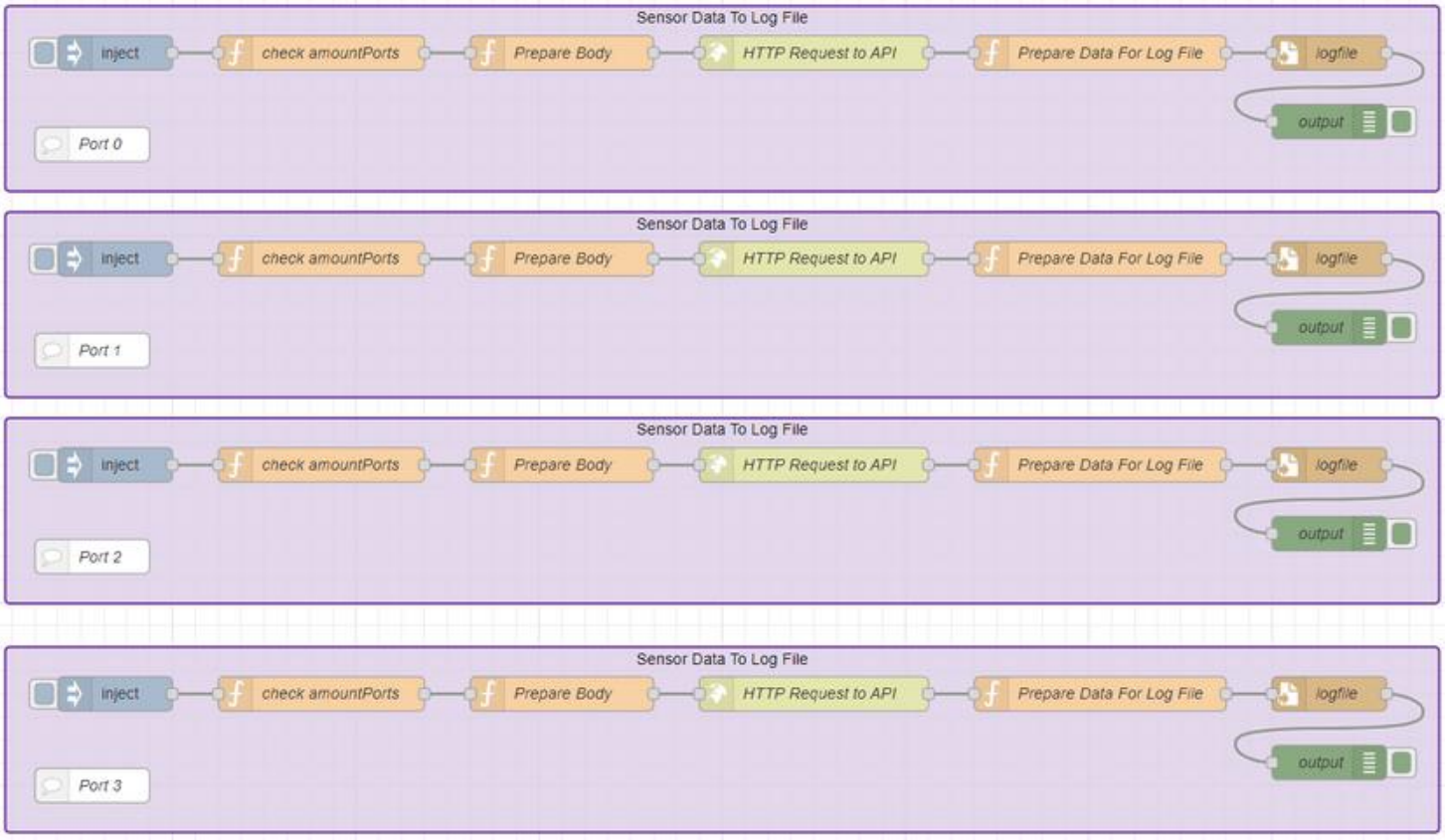
Extra informatie over VBM_AMOUNT_PORTS:

De SIG200 API werkt met poortnummers 0, 1, 2 en 3. De omgevingsvariabele VBM_AMOUNT_PORTS werkt met 1, 2, 3, 4. Dit is leesbaarder.
Dus als je slechts 3 sensoren wilt aansluiten, wijs je VBM_AMOUNT_PORTS = 3 toe.

Waarom VBM_AMOUNT_PORTS gebruiken?

Als we niet 4 punten hebben om te meten, en we hebben bijvoorbeeld maar 2 of 3 sensoren nodig (voor 2 of 3 punten). Deze sensoren kunnen duur zijn, dus in onze oplossing moeten we daar rekening mee houden.
We kunnen VBM_AMOUNT_PORTS gelijkstellen aan 2 of 3 en dan doet de andere poort niets. Het genereert ook geen fouten, het is gewoon inactief.

9.2 Node-RED flow



Figuur 16: Node-RED Flow

Inject: We gebruiken de omgevingsvariabele VBM_SIG_IP in de injectie-node zodat we deze later kunnen gebruiken in de HTTP-verzoek-node. Hier zorgen we er ook voor dat de verbonden nodes worden getriggerd.

De frequentie waarmee de API wordt aangeroepen:

De injectie-nodes in Node-RED worden elke 1,4 seconden getriggerd. Met deze snelheid krijgt de API niet te veel oproepen en geeft hij geen fouten terug.

Als we 4 sensoren hebben aangesloten, hebben we ongeveer 45x4 API-oproepen per minuut.

Op deze manier hebben we meer dan genoeg gegevens en wordt het systeem niet overbelast. Er is geen one-size-fits-all, maar het is perfect voor onze oplossing.

Check amountPorts:

```
var amountPorts = env.get("VBM_AMOUNT_PORTS"); // Retrieve the value of amountPorts from the environment variable and convert it to an integer

// Check if amountPorts meets certain conditions
if (amountPorts <= 4) {
    // If the conditions are met, pass the message on to the next part of the flow
    return msg;
} else {
    // If the conditions are not met, stop the message and ensure it does not proceed to the next nodes
    return null;
}
```

Deze functienode wordt gebruikt om te controleren wanneer een poort moet worden uitgelezen.

Bijvoorbeeld: we hebben slechts 3 punten om te meten, dus hebben we slechts 3 sensoren nodig. We stellen VBM_AMOUNT_PORTS in op 3. Op deze manier worden alleen poort 0, 1 en 2 uitgelezen. Poort 3 wordt niet uitgelezen.

Prepare Body function:

```
msg.headers = {};
msg.headers['Content-Type'] = 'application/json';
//
msg.payload = JSON.parse('{ "header": { "portNumber": 0}, "data": { "processData": "in" }} ');
return msg;
```

Deze functie bereidt de body voor om te worden verzonden met het POST-request. Het "portNumber" is 0, 1, 2 of 3.

HTTP Request to API:

Hier maken we het POST-verzoek naar de API en we gebruiken de VBM_SIG_IP omgevingsvariabele voor het IP-adres.

Prepare Data For Log File:

```
// Assuming msg.payload contains the response from the HTTP request
```

```
//rmsX
var Byte18 = msg.payload.data.processDataIn["18"];
//rmsY
var Byte19 = msg.payload.data.processDataIn["19"];
//rmsZ
var Byte20 = msg.payload.data.processDataIn["20"];

// Create an array containing the values from keys "18", "19",
and "20"
var ByteArray = [Byte18, Byte19, Byte20];

//make a timestamp
const currentDate = new Date();
var currentTime = currentDate.getTime()
//log message
var loggingMessage = {
    "vibration-data": {
        "sig200IP": env.get("VBM_SIG_IP"),
        "portNumber": 0,
        "timestamp": currentTime,
        "rmsX": Byte18,
        "rmsY": Byte19,
        "rmsZ": Byte20,
        "smartbox": env.get("DEVICE_NAME"),
        "logfile": env.get("VBM_LOG_FILE")
    }
}
//turn the json object into a string
loggingMessage = JSON.stringify(loggingMessage)
msg.payload = loggingMessage

return msg;
```

We bereiden de gegevens voor zodat ze naar het logbestand (vibration-*.log) kunnen worden verzonden. Byte 18, 19 en 20 zijn de bytes van de v-RMS (of a-RMS) gegevens.

Logfile:

Op de pi wordt de logfile opgeslagen onder de naam “vibration-*.log” (* kan 1 of 2… zijn, afhankelijk van environment variabele).

Voorbeeld logline:

"{"vibration-data":{"sig200IP":"10.1.61.100","portNumber":2,"timestamp":1715153310672,"rmsX":0.129001617,"rmsY":0.181105184,"rmsZ":0.21392951,"logfile":"/home/pi/log_node_red/vibration-1.log"}}"

10. Logfile Analyzer

Onze Node-RED genereert een logbestand dat de trillingsgegevens bevat. Omdat we elke 1,4 seconden meten, kan het bestand erg groot worden.

Om dit probleem op te lossen implementeren we een logbestand-analyseservice die continu de opgegeven logbestanden controleert. Als het logbestand te groot wordt, worden de oudste X regels verwijderd.
Op deze manier houden we ons vibration-*.log-bestand klein en neemt het niet te veel ruimte in op de Pi.

Opmerking: De logbestand-analyseservice kan worden gebruikt voor elk logbestand, maar moet altijd actief en geïnstalleerd zijn als je trillingen meet
Als je dit niet installeert, kan het bestand na een paar weken te groot worden en moet je het bestand handmatig verwijderen.

Code:

```
#!/usr/bin/env python3
# Variables for specifying the maximum number of lines to keep and
the total number of lines
max_lines_to_keep = 500
total_lines_threshold = 1000

import os
import glob
import time

def keep_recent_lines(log_file_path, max_lines=max_lines_to_keep,
total_lines=total_lines_threshold):
    try:
        # Read the current log data
        with open(log_file_path, 'r') as file:
            log_data = file.readlines()

        # Keep only the most recent lines if there are more than the
total_lines threshold
        if len(log_data) > total_lines:
            log_data = log_data[-max_lines:]

            # Write the updated log data back to the log file
            with open(log_file_path, 'w') as file:
                file.writelines(log_data)
    except Exception as e:
        print(f"An error occurred while processing {log_file_path}:
{e}")
```

```
def clean_logs_in_folder(folder_path):
    # Get a list of all log files in the folder
    log_files = glob.glob(os.path.join(folder_path, '*.log'))

    # Iterate over each log file in the folder
    for log_file_path in log_files:
        # Keep the most recent lines from each log file
        keep_recent_lines(log_file_path)

def clean_logs_in_files_and_folders(paths):
    # Iterate over each path
    for path in paths:
        if os.path.isfile(path):
            # If the path is a file, clean it
            keep_recent_lines(path)
        elif os.path.isdir(path):
            # If the path is a directory, clean all log files within
it
            clean_logs_in_folder(path)
        elif '*' in path:
            # If the path contains wildcard characters, use glob to
match files
            matching_files = glob.glob(path)
            for file_path in matching_files:
                keep_recent_lines(file_path)

if __name__ == "__main__":
    # Define a list of log file paths, including individual files,
folders, and paths with wildcard characters
    log_paths = [
                '/home/pi/log_node_red/vibration-1.log',
                '/home/pi/log_node_red',
                '/tmp/log',
                # Add more log file paths or folder paths with
wildcard characters here if needed
    ]

    # Main loop to run the script periodically
    while True:
        # Clean log files in individual files, folders, and paths
with wildcard characters
        clean_logs_in_files_and_folders(log_paths)

        # Pause execution for 10 seconds
        time.sleep(10)
```

Ansible voor logfile analyzer:

```
max_lines_to_keep: 500
total_lines_threshold: 1000
log_files:
  - '/home/pi/log_node_red/vibration-1.log'
  - '/home/pi/log_node_red'
  - '/tmp/log'

  # Add more log file paths or patterns here if needed
```

11. Ansible

Figuur 17: Ansible Logo

Ansible is een open-source automatiseringsplatform dat wordt gebruikt voor het automatiseren van IT-taken zoals configuratiebeheer, applicatie-implementatie, netwerkautomatisering en orchestratie van IT-workflows. Met Ansible kunnen beheerders taken automatiseren door gebruik te maken van eenvoudige, op YAML gebaseerde playbook-scripts, die de gewenste staat van een systeem beschrijven. Het biedt een eenvoudige en krachtige manier om complexe IT-omgevingen te beheren en te automatiseren.

Met Ansible kunnen we makkelijk alle pi's in de smartboxen configureren.

11.1 Ansible Files

Hier wordt beschreven wat er allemaal in de Ansible structuur is aangepast zodat de nieuwe vibratie-functionaliteit kan worden toegevoegd aan de smartbox.

Ansible-sb-default
playbook.yaml:

```
# # # # #
##########################################
##########################################
# # # # # #########ALL THE FUNCTIONALITY'S THAT CAN BE PUT ON A SMARTBOX###########
# # # # #
##########################################
##########################################
- import_playbook:  ../ansible-role-functionality/vibration.yaml
```

In "ansible-sb-default", is de nieuwe functionaliteit toegevoegd (vibration.yaml).

Ansible-role-functionality
vibration.yaml:

```
---

- hosts: "{{ server_hosts }}"
  # remote_user: pi
  vars:
    node_red_port: 1600
    node_red_folder: vibration
    node_red_folder_path: /home/pi/.node-red/
    env_vars_path: /home/pi/.node-red/env_variables
```

```
    necessary_env:
      - "{{ env }}"
  become: true
  roles:
    - role: functionality
...
```

De nieuwe vibratie-functionaliteit staat op poort 1600.

Node-red-functionality-files
/variables/sb21-0183:
DEVICE_NAME=sb21-0183
DEVICE_NAME_FULL=sb21-0183.rad.priv.vangenechten.com
DEVICE_MAC= mac van pi
ENV=rad
VBM_SIG_IP= hier komt het ip
VBM_AMOUNT_PORTS=4
VBM_LOG_FILE=/home/pi/log_node_red/vibration-1.log

Hier vinden we de environment variables "VBM_SIG_IP", "VBM_AMOUNT_PORTS" en VBM_LOG_FILE.

/flows/vibration-flows.json
Deze file bevat de JSON code die de Node-RED flow maakt.

12. Filebeats

Figuur 18: Filebeat Logo

Filebeat is een lichtgewicht, door Elastic ontwikkeld programma voor het verzamelen en doorsturen van logbestanden. Het is ontworpen om logs van verschillende bronnen te verzamelen, zoals bestanden op lokale systemen of logboekuitvoer van applicaties, en deze door te sturen naar een centrale locatie, zoals Elasticsearch of Logstash, voor analyse en visualisatie. Met Filebeat kunnen gebruikers logdata efficiënt en betrouwbaar verzamelen en doorsturen, waardoor het proces van logbeheer en monitoring wordt gestroomlijnd. Het is vooral handig in gedistribueerde omgevingen waar veel loggegevens worden gegenereerd en gecentraliseerd moeten worden voor analyse en troubleshooting.

Hier wordt beschreven wat er allemaal is aangepast in de filebeat configuratie.

filebeats.yml:

```
# ============================== Filebeat inputs
===============================
- type: filestream
  id: vibrationfilestream
  enabled: true
  paths:
    - /home/pi/log_node_red/vibration*.log
  parsers:
    - ndjson:
        target: ""
        message_key: vibration-data
  tags: ["vibrationdata"]
```

Om de gegevens uit het logbestand vibration-*.log naar Kafka te krijgen, gebruiken we Filebeat. We hebben een nieuwe filestream-input toegevoegd om de trillingsgegevens naar het onderwerp "smartboxlogging" te sturen, waar alle smartbox-gegevens worden bewaard.

In Kafka komen deze gegevens binnen.

De tag is [vibrationdata].

Figuur 19: JSON Kafka

13. Logstash

logstash

Figuur 20: Logstash Logo

Logstash is een open-source logverzamelings- en -verwerkingstool die wordt gebruikt voor het verzamelen, transformeren en laden (ETL) van loggegevens in verschillende formaten en van verschillende bronnen. Het kan loggegevens van diverse bronnen, zoals logbestanden, databases, systeemlogboeken en meer, verzamelen en deze vervolgens transformeren en verrijken voordat ze worden doorgestuurd naar een eindbestemming, zoals Elasticsearch, Kafka of een ander analyseplatform. Met Logstash kunnen gebruikers complexe pijplijnen maken om loggegevens te analyseren, te indexeren en te visualiseren, waardoor het proces van logbeheer en monitoring wordt gestroomlijnd en vereenvoudigd.

Hier wordt beschreven wat er allemaal in logstash is verandert om de gegevens te versturen:

/etc/logstash/conf.d/smartboxlogging.conf:

Filter:

```
if "vibrationdata" in [tags]{
     mutate {
     remove_field => ["[event][original]"]
     }
   }
```

Output:

```
if "vibrationdata" in [tags]{
      elasticsearch {
          hosts =>
["https://elasticsearch01.rad.priv.vangenechten.com:9200"]
          ssl => true
          ssl_certificate_verification => false
          cacert => "/etc/ssl/logstash01/rad_bundle.crt"
          manage_template => true
```

```
        index => "vibrationdata"
        user => "elastic"
        password => "UENJH@Cf?ikzpd7q"
        document_id => "%{[@metadata][_id]}"
        }
}
```

In "smartboxlogging.conf", is er een nieuwe filter en output om de trillingsgegevens naar Elasticsearch te krijgen. De filter is bedoeld om te voorkomen dat dubbele gegevens naar Elasticsearch worden verzonden.

14. Elasticsearch

Figuur 21: Elasticsearch Logo

Elasticsearch is een open-source, gedistribueerd zoek- en analyse-engine, gebaseerd op Apache Lucene. Het wordt veel gebruikt voor het doorzoeken, analyseren en visualiseren van grote hoeveelheden gestructureerde en ongestructureerde gegevens in realtime. Elasticsearch wordt vaak gebruikt voor het bouwen van zoekmachines, logboekanalyse, monitoring van infrastructuur, en het uitvoeren van complexe analyses op gegevens. Het biedt krachtige zoekmogelijkheden, waaronder full-text search, aggregaties, filtering, en geospatiale zoekopdrachten.

Onze vibratiedata wordt opgeslagen in elasticsearch database onder de index “vibrationdata”.

15. Grafana

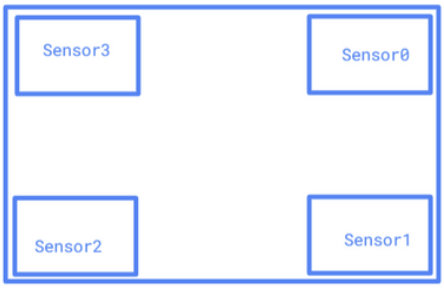
Figuur 22: Grafana Logo

Grafana is een visualisatie en monitoringstool.

We gebruiken Grafana om onze trillingsgegevens te visualiseren. Op basis van deze visualisaties willen we voorspellingen maken, trends analyseren...
Er zijn veel verschillende manieren om onze gegevens te visualiseren, maar ze zijn niet allemaal relevant. Hieronder wordt uitgelegd welk type grafieken (groepen) we hebben gemaakt en waarom we ze hebben gemaakt.

15.1 Algemene Uitleg

We kunnen tot 4 sensoren aansluiten op de SIG200. De poortnummers van de SIG200 zijn 0,1,2 en 3. Daarom gebruiken we de namen Sensor0 (voor poort 0), Sensor1 (voor poort 1), Sensor2 (voor poort 2) en Sensor3 (voor poort 3).

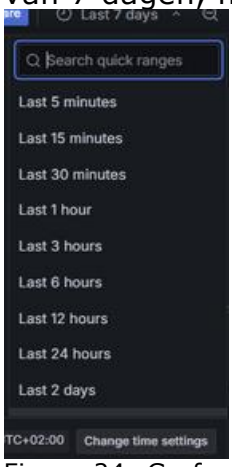


Figuur 23: Airco Sensor Installatie

De grafieken zijn ontworpen om zowel gegevens van alle sensoren gecombineerd als individuele sensorgegevens weer te geven.

In Grafana worden de gegevens gegroepeerd en geanalyseerd op basis van de geselecteerde tijdsperiode (30 minuten, 1 uur, 7 dagen...). Bijvoorbeeld, bij het berekenen van statistieken zoals gemiddelden of pieken, aggregeert Grafana alle individuele metingen binnen het gespecificeerde tijdsinterval.

De onderstaande grafieken zijn gemaakt met een tijdsperiode van 7 dagen, maar dit kan eenvoudig worden aangepast:



Figuur 24: Grafana Tijdfilters

15.2 Graph Group 1: Average Vibrations

Figuur 25: Graph Group 1

- Gemiddelde Sensor Trillingen X-as + individueel:
  Deze grafiek visualiseert de gemiddelde trillingen op de X-as. De 'Average Sensor Vibrations (groen)' vertegenwoordigen de gemiddelde trillingen gemeten door alle sensoren op de X-as.
  De 'Sensor0-Sensor3 Average (geel, blauw, oranje, rood)' geven de gemiddelden weer van de individuele sensoren op de X-as.

- Gemiddelde Sensor Trillingen Y-as + individueel:
  Deze grafiek visualiseert de gemiddelde trillingen op de Y-as. De 'Average Sensor Vibrations (groen)' vertegenwoordigen de gemiddelde trillingen gemeten door alle sensoren op de Y-as.
  De 'Sensor0-Sensor3 Average' lijnen geven de gemiddelden weer van de individuele sensoren op de Y-as.

- Gemiddelde Sensor Trillingen Z-as + individueel:
  Deze grafiek visualiseert de gemiddelde trillingen op de Z-as. De 'Average Sensor Vibrations (groen)' vertegenwoordigen de gemiddelde trillingen gemeten door alle sensoren op de Z-as.
  De 'Sensor0-Sensor3 Average' lijnen geven de gemiddelden weer van de individuele sensoren op de Z-as.

- Gemiddelde Sensor Trillingen XYZ-as:
  Deze grafiek visualiseert de gemiddelde trillingen op de XYZ-as.

Waarom zijn gemiddelde trillingen nuttig?
Het meten van gemiddelde trillingen biedt een uitgebreid overzicht van de algemene trillingsniveaus, waardoor de detectie van algemene trends in de loop van de tijd mogelijk is. Het geeft ons een algemeen beeld van de machine trillingen. Zo kan bijvoorbeeld het waarnemen van een toename van de gemiddelde trillingsniveaus over een periode, zoals een maand, wijzen op mogelijke problemen zoals toenemende slijtage of groeiende onbalans binnen de machines.

Voorbeeld
Gedurende een observatieperiode van drie maanden vertoonde de machine consistent gemiddelde trillingen rond 0.3 bij normaal gebruik. Over de laatste twee dagen hebben we een stijging in trillingen opgemerkt, met niveaus rond 0.8. Deze plotselinge verandering suggereert dat er mogelijk iets mis is met de werking van de machine en verdient nader onderzoek. Het detecteren van dergelijke verschuivingen in trillingspatronen helpt ons potentiële problemen vroegtijdig op te sporen, waardoor we proactieve maatregelen kunnen nemen om storingen of onderbrekingen van de apparatuur te voorkomen.

15.3 Graph Group 2: Peak Vibrations

Figuur 26: Graph Group 2

- Pieken in trillingsniveaus op de X-as + individueel:
  Deze grafiek visualiseert de piektrillingen op de X-as. De 'Peak Vibrations (groen)' vertegenwoordigen de piektrillingen gemeten door alle sensoren op de X-as. De 'Sensor0-Sensor3 Peaks (geel, blauw, oranje, rood)' geven de pieken weer van de individuele sensoren op de X-as.

- Pieken in trillingsniveaus op de Y-as + individueel:
  Deze grafiek visualiseert de piektrillingen op de Y-as. De 'Peak Vibrations (groen)' vertegenwoordigen de piektrillingen gemeten door alle sensoren op de Y-as. De 'Sensor0-Sensor3 Peaks (geel, blauw, oranje, rood)' geven de pieken weer van de individuele sensoren op de Y-as.
- Pieken in trillingsniveaus op de Z-as + individueel:
  Deze grafiek visualiseert de piektrillingen op de Z-as. De 'Peak Vibrations (groen)' vertegenwoordigen de piektrillingen gemeten door alle sensoren op de Z-as. De 'Sensor0-Sensor3 Peaks(geel, blauw, oranje, rood)' geven de pieken weer van de individuele sensoren op de Z-as.

- Pieken in trillingsniveaus op de XYZ-as:
  Deze grafiek visualiseert de piektrillingen op de XYZ-as.

<u>Waarom zijn piektrillingen nuttig?</u>
Het analyseren van piektrillingen biedt waardevolle inzichten in kritieke gebeurtenissen die zich kunnen voordoen binnen machines. Pieken in trillingsniveaus kunnen wijzen op verschillende significante gebeurtenissen zoals overbelasting, impact, structurele problemen, of het begin van opkomende problemen die zich intermitterend manifesteren met hoge intensiteit. Bijvoorbeeld, een piek in trillingsniveaus zou kunnen wijzen op het optreden van een storing resulterend in verhoogde trillingen, of een incident dat plotselinge impact-geïnduceerde trillingen veroorzaakt.

Voorbeeld
Laten we zeggen dat we de piektrillingen van een machine de afgelopen paar maanden hebben gemonitord. Typisch bleven de piektrillingen onder de 1.0, wat wijst op normale werking. Echter, in de afgelopen week hebben we verschillende gevallen waargenomen waarbij de piektrillingen de 2.0 overschreden. Deze significante toename in piektrillingen duidt op een afwijking van het gebruikelijke gedrag en suggereert dat er mogelijk een probleem is met de machine.

15.4 Graph Group 3: Heatmaps of Average

Figuur 27: Graph Group 3

- Sensor0 XYZ:
  Deze grafiek toont de gemiddelde trillingen van Sensor0 op de XYZ-as.

- Sensor1 XYZ:
  Deze grafiek toont de gemiddelde trillingen van Sensor1 op de XYZ-as.

- Sensor2 XYZ:
  Deze grafiek toont de gemiddelde trillingen van Sensor2 op de XYZ-as.

- Sensor3 XYZ:
  Deze grafiek toont de gemiddelde trillingen van Sensor3 op de XYZ-as.

- Gemiddelde Sensor Trillingen XYZ-as:
  Deze grafiek toont de gemiddelde trillingen van alle sensoren op de XYZ-as.

<u>Waarom zijn heatmaps nuttig?</u>
Heatmaps bieden een visuele representatie van trillingspatronen, waardoor eenvoudige identificatie van patronen en afwijkingen op basis van hun intensiteit mogelijk is. Door waarden te kleuren op basis van hun magnitude, vergemakkelijken heatmaps de identificatie van problematische gebieden binnen machines. Deze visuele tool vereenvoudigt het proces van het vinden van potentiële problemen. Bovendien vergemakkelijken heatmaps vergelijkingen tussen verschillende machines, waardoor waardevolle inzichten worden verkregen in hun operationele prestaties. Door numerieke gegevens om te zetten in kleurgradaties, verbeteren heatmaps de efficiëntie van trillingsanalyse.

Voorbeeld
In de afgelopen drie maanden vertoonde onze heatmaps meestal kleuren zoals groen, oranje en soms roze. Maar als we plotseling een grote paarse vlek op de heatmap zien, is dat zorgwekkend. Het kan betekenen dat er iets ongebruikelijks aan de hand is. Dit zou een groot probleem kunnen zijn dat direct moet worden opgelost. Het kan aangeven dat de machines te veel trillen of niet goed werken.

16. Predictive Maintenance Blueprint

Hoe doen we nu het beste predictive maintenance? Dit wordt uitgelegd in deze “Predictive Maintenance Blueprint”.

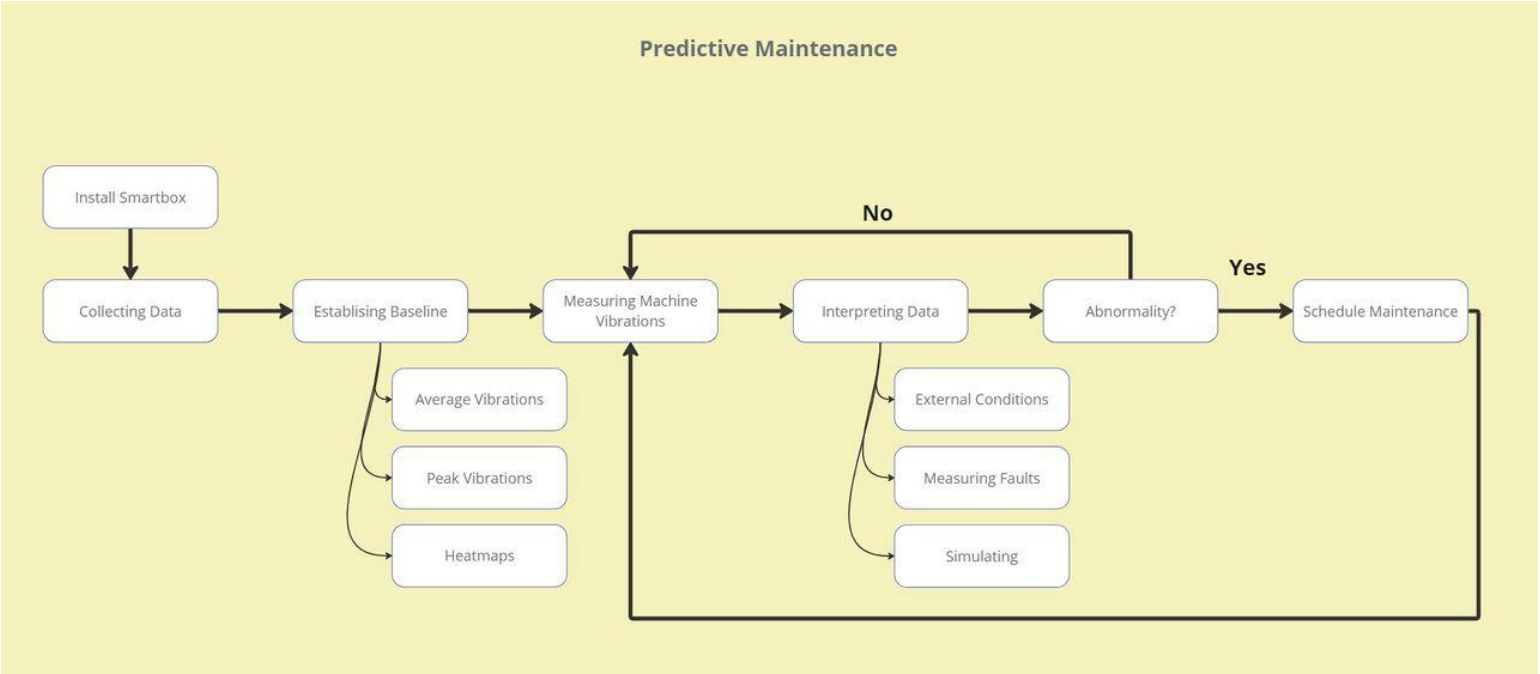
16.1 Introductie

Dit deel van de documentatie dient als een gids voor het implementeren van predictief onderhoud op basis van trillingsanalyse. Het biedt een framework voor degenen die streven naar het optimaliseren van apparatuuronderhoud door middel van proactieve monitoring.

16.2 Doel

Het Blueprint voor Predictive Maintenance schetst essentiële stappen voor het benutten van trillingsgegevens om apparatuurstoringen te anticiperen en onderhoudsactiviteiten effectief in te plannen. Hoewel het mogelijk niet elk detail behandelt, biedt het waardevolle inzichten om toekomstige projecten te inspireren en te informeren.

16.3 Flowchart



Figuur 28: Flowchart

16.4 Predictive Maintenance met Vibraties

Predictive maintenance is een proactieve benadering van apparatuuronderhoud die tot doel heeft om storingen te anticiperen en te voorkomen voordat ze zich voordoen. Een manier om dit te doen is door trillingen te meten. Op deze manier kunnen we de algemene trillingspatronen, veranderingen in algemene trends, pieken... zien.
Net zoals een arts de hartslag van een patiënt controleert om hun gezondheid te beoordelen, kunnen ingenieurs en onderhoudsprofessionals trillingen monitoren om de "gezondheid" van de machine te beoordelen. We kunnen naar de trillingen kijken alsof het de hartslag van een machine is. Vergelijkbaar met hoe een hartslag continue feedback geeft over de toestand van het hart, bieden trillingen continue inzichten in de toestand van machines. Deze trillingen worden gegenereerd door verschillende mechanische onderdelen zoals motoren, lagers, tandwielen... terwijl ze werken. En net zoals elke persoon een uniek hartslagpatroon heeft, hebben verschillende soorten machines specifieke trillingskenmerken die geassocieerd worden met hun normale werking.

Enkele voordelen van predictive maintenance:

• Vroegtijdige foutdetectie: door subtiele veranderingen in trillingspatronen, pieken... te detecteren

• Conditiemeting-gebaseerd onderhoud: in plaats van vaste schema's te volgen, kunnen onderhoudsactiviteiten worden gepland op basis van de werkelijke conditie van de apparatuur, waardoor de allocatie van middelen wordt geoptimaliseerd en de downtime wordt geminimaliseerd.

• Verbeterde betrouwbaarheid van apparatuur: door de fouten te spotten voordat ze optreden

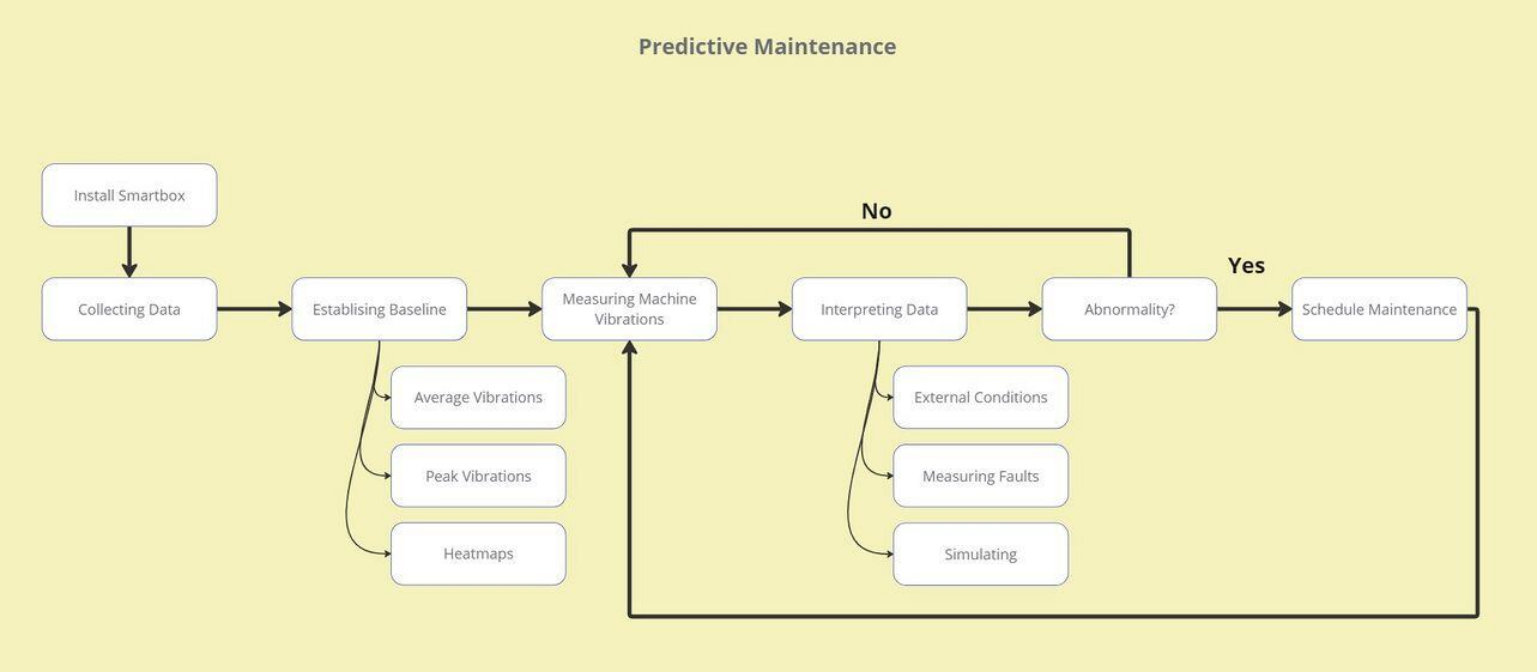
• Kostenbesparingen: door ongeplande stilstand te vermijden, kunnen organisaties aanzienlijke kostenbesparingen realiseren in de loop van de tijd

16.5 Het idee

We gaan predictive maintenance uitvoeren met behulp van een “baseline”. De baseline biedt een referentiepunt dat normale bedrijfsomstandigheden vertegenwoordigt. Door de huidige gegevens te vergelijken met deze baseline, kunnen afwijkingen en anomalieën worden geïdentificeerd, wat wijst op mogelijke problemen met de apparatuur.

Zonder een baseline zou het moeilijk zijn om te weten wat de gegevens betekenen.

16.6 Flowchart uitleg



Figuur 29: Flowchart

Collecting Data

Nu we weten wat predictive maintenance is, kunnen we beginnen met de belangrijkste stap: het verzamelen van gegevens. Met deze gegevens kunnen we een baseline maken, voorspellingen doen, AI-modellen trainen, trends opsporen...
De gegevens die worden verzameld, moeten afkomstig zijn uit een periode waarin de machine onder normale omstandigheden functioneerde. Er mogen geen storingen, fouten, onderhoud... zijn geweest. Het doel is om betrouwbare gegevens te verkrijgen over de normale werking van de machine.
De hoeveelheid tijd die nodig is om gegevens te verzamelen, is afhankelijk van de machine en de eisen van het project. Er zijn een paar factoren om rekening mee te houden:

- Tijd en budget: hoeveel tijd heb je voor je project en wat is het budget?
- Operationele cyclus: wat is de operationele cyclus van de machine (opstarten, werking, uitschakelen...)
- Omvang en complexiteit van de apparatuur: grotere of complexere apparatuur kan langere periodes van gegevensverzameling vereisen om een uitgebreid scala aan trillingspatronen vast te leggen

...

Er is geen one-size-fits-all antwoord op deze vraag, het doel is altijd om betrouwbare gegevens te hebben over de normale werking van de machine.

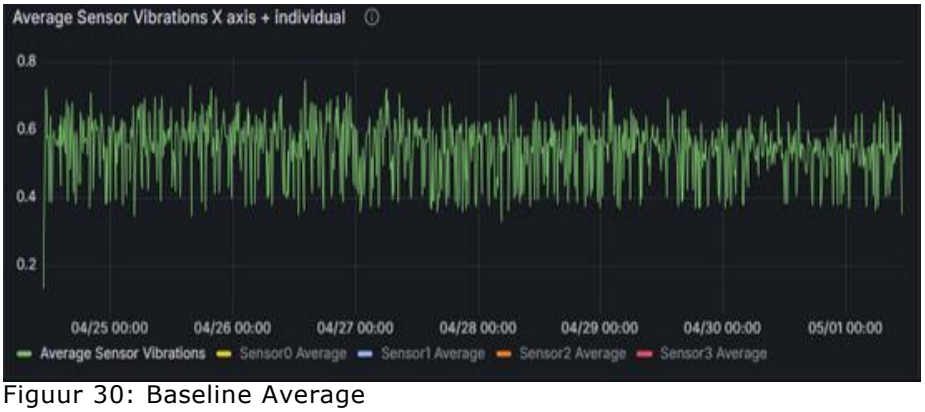
Voorbeeld airconditioningunit (AC):
Laten we het voorbeeld nemen van een airconditioningunit (AC) in een gebouw. De AC start elke ochtend om 8:00 uur en stopt om 18:00 uur, en werkt dit schema elke dag van de week.

De eerste stap is het verzamelen van trillingsgegevens over de AC-unit terwijl deze onder normale omstandigheden werkt. Dit betekent ervoor zorgen dat de AC-unit vrij is van storingen, fouten of onderhoudsinterventies tijdens de periode van gegevensverzameling. De AC-unit moet tijdens deze periode normaal functioneren. Zo zou hij geen uitval mogen hebben gehad vanwege storingen op een woensdag om 13:00 uur, noch zou er onderhoud mogen zijn uitgevoerd op een donderdag tussen 10:00 en 15:00 uur. Het airconditioningsysteem moet gedurende de hele gemeten duur normaal hebben gewerkt, beginnend om 8:00 uur, zijn taken uitvoeren en stoppen om 18:00 uur.
Nu, voor de duur van de gegevensverzameling, willen we minstens 1 volledige operationele cyclus (8:00-18:00). In theorie is 1 cyclus genoeg omdat je nu verschillende cycli kunt gaan vergelijken. Maar het is waarschijnlijk beter om minstens een paar dagen van de normale werking van de airco te hebben (meer cycli) of zelfs 1 week, afhankelijk van je tijd / budget en de complexiteit van de airco (als het een zeer grote is).

Establishing Baseline

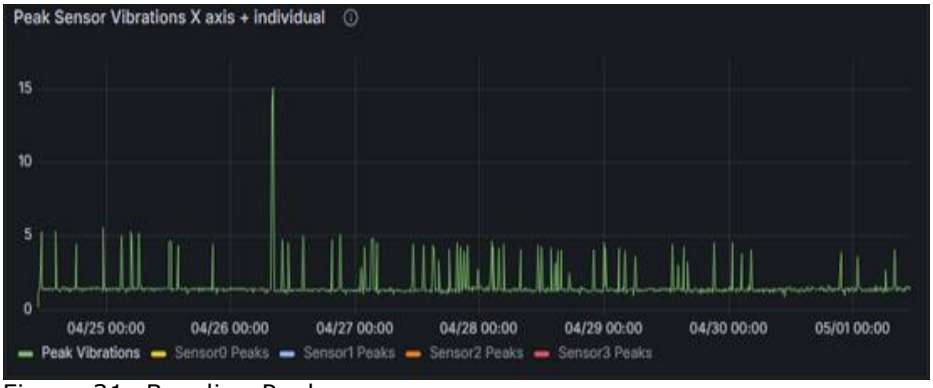
Dus zodra we onze gegevens hebben verzameld, kunnen we een baseline maken. Een baseline is een grafiek gemaakt van gegevens wanneer de machine normaal functioneerde, zodat je later de huidige gegevens kunt vergelijken met deze " baseline " grafiek.
Deze baseline dient als referentiepunt voor het detecteren van afwijkingen of abnormale patronen in toekomstige gegevens, wat wijst op mogelijke problemen met de apparatuur.
Zoals uitgelegd op deze wiki, zijn de gemiddelde, piek- en heatmap-grafieken altijd goede opties. Daarom zou je altijd een "gemiddelde baseline grafiek", "peak baseline grafiek" en "heatmap baseline grafiek" moeten maken.

- Voorbeeld “Gemiddelde Baseline Grafiek”



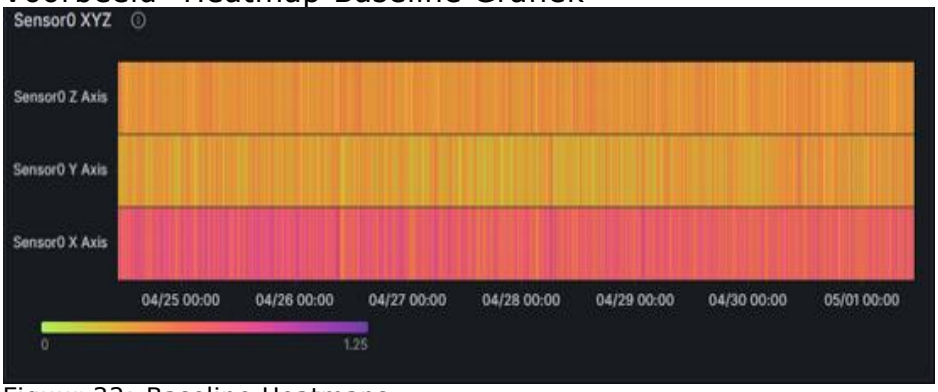
Figuur 30: Baseline Average

- Voorbeeld “Peak Baseline Grafiek”



Figuur 31: Baseline Peak

- Voorbeeld “Heatmap Baseline Grafiek”



Figuur 32: Baseline Heatmaps

Voorbeeld airconditioningunit (AC):
We meten een week lang normale operationele cycli van de AC. We verzamelen deze gegevens en maken met deze gegevens grafieken die dienen als baseline, zodat we de huidige gegevens kunnen vergelijken met deze "baseline" grafieken.
De bovenstaande grafieken zouden voorbeeld baselinegrafieken kunnen zijn. Vervolgens vergelijk je je nieuwe "live" grafieken met deze baselinegrafieken. Je kunt veranderingen in kleur opmerken, meer en hogere pieken, een hoger gemiddelde...

Interpreting Data

Nadat we onze baseline hebben gemaakt, kunnen we de huidige gegevens vergelijken met deze baseline. Zonder dit zou het moeilijk zijn om te begrijpen wat de gegevens betekenen.

Voorbeeld airconditioningunit (AC):
Bij het meten van de operationele cyclus van de AC (tijdens normaal gebruik) zien we dat de gemiddelde vibraties altijd rond de 0,5 meten. Maar de laatste 2 dagen zien we dat de gemiddelde vibraties langzaam stijgen van 0,5 naar 0,6 naar 0,7 naar 0,9. Omdat we kunnen vergelijken met onze baseline, waar het altijd rond de 0,5 vibreerde, kunnen we nu bepalen dat er mogelijk een probleem is en onderhoud inplannen.
Of als we op de baseline van de pieken zien dat een piek van 1,2 of zelfs 3 normaal is, maar plotseling zien we veel pieken die boven de 5 gaan.
Hetzelfde kan worden gedaan met warmtekaarten. Plotse kleurveranderingen kunnen wijzen op mogelijke problemen.

Het interpreteren van gegevens kan moeilijk zijn. We moeten weten wat de gegevens betekenen, wat normaal is, wat niet normaal is, omgevingsfactoren...

Hier zijn enkele dingen om over na te denken:
• Externe omstandigheden
Wat als er heftrucks voorbijrijden? Wat als er onderhoud wordt gepleegd aan een nabijgelegen machine? Wat als er veel activiteit is in de buurt van de machine?
Er zijn veel externe omstandigheden die de trillingen kunnen beïnvloeden.
Gelukkig zijn er enkele oplossingen:
-Kalibratie en filtering van sensorgegevens: Implementeer kalibratieprocedures om ervoor te zorgen dat sensoren nauwkeurige metingen leveren, zelfs onder invloed van externe trillingen veroorzaakt door bijvoorbeeld passerende heftrucks. Gebruik geavanceerde filtertechnieken om externe ruis te minimaliseren en de kwaliteit van verzamelde gegevens te verbeteren.
-Contextuele analyse: Neem omgevingsfactoren zoals heftruckverkeer, nabijgelegen bouwactiviteiten en andere omgevingsvariabelen op in de analyse van sensorgegevens. Als bijvoorbeeld een heftruck altijd dezelfde trillingen veroorzaakt, en we dat patroon op de grafiek opmerken, kunnen we het negeren.

• Fouten meten
Het meten van fouten is een nuttige tactiek om betekenis te geven aan gegevens. Het stelt ons in staat om gemeenschappelijke problemen of afwijkingen die zich kunnen voordoen in de machines te identificeren. Door systematisch deze fouten te meten en te analyseren, kunnen we een dieper inzicht ontwikkelen in hun kenmerken en patronen, waardoor het gemakkelijker wordt om ze te herkennen in toekomstige gegevens.

Sommige machines kunnen gemeenschappelijke fouten hebben zoals lager slijtage, verkeerde uitlijning of onbalans. Als we deze fouten meten wanneer ze zich voordoen, kunnen we zien hoe het eruitziet op de grafieken. Er kunnen enkele patronen zijn die we kunnen zien, hoelang het duurt... Door de typische patronen te begrijpen die gepaard gaan met elke fout, kunnen we de afwijkingen herkennen en actie ondernemen.
Bijvoorbeeld, een plotselinge toename van trillingsniveaus bij een specifieke frequentie kan duiden op lagerslijtage, terwijl periodieke pieken in trillingsniveaus kunnen wijzen op mechanische speling.

• Simulatie
Een andere nuttige tactiek om gegevens beter te begrijpen is door fouten en externe omstandigheden te simuleren.
-Fouten simuleren: we bootsen het gedrag na dat geassocieerd is met specifieke fouten. We kunnen bijvoorbeeld opzettelijk lagerslijtage of verkeerde uitlijning induceren in een testopstelling en de resulterende veranderingen in trillingspatronen observeren.
-Externe omstandigheden: we kunnen omgevingsfactoren repliceren die de machine trillingen kunnen beïnvloeden. We kunnen bijvoorbeeld trillingen of verstoringen introduceren in de testomgeving om de effecten van nabijgelegen machines of voertuigverkeer, zoals een voorbijrijdende heftruck, te simuleren.

Het is niet eenvoudig om betekenis te geven aan gegevens. Het is een lang proces dat uitgebreide gegevensverzameling, analyse en experimenten vereist. Hoewel gegevens waardevolle in

16.7 Conclusie

Als je deze blueprint voor predictive maintenance volgt, heb je een solide basis. Het principe blijft hetzelfde. We beginnen met het verzamelen van gegevens, we maken een baseline en dan kunnen we live gegevens vergelijken met deze baseline. Het moeilijke is om te begrijpen wat de gegevens betekenen en alle factoren die meespelen. Dit is iets wat je gaandeweg zult ontdekken. Als je genoeg gegevens hebt over het normale functioneren van de machine, maar ook over typische fouten die optreden (door simulatie of gewoon meten), kun je een soort plan maken om deze fouten van tevoren te herkennen en te voorkomen. Hetzelfde geldt voor externe factoren die trillingen beïnvloeden. Je zou ook een filter kunnen ontwikkelen om al het extra geluid (trillingen) te elimineren.
Door je aanpak voortdurend te verfijnen, kun je de efficiëntie en betrouwbaarheid van je apparatuur maximaliseren terwijl je onverwachte downtime minimaliseert.

17. Hoe presenteren we dit idee aan anderen (mensen in de fabriek)?

De werknemers in de fabriek moeten ervan overtuigd worden dat deze oplossing voor predictive maintenance echt kan werken en niet zal ingrijpen in de huidige processen en situatie. De beste manier om dit te doen is door het uit te leggen, ze een algemeen overzicht te geven van onze oplossing en hoe deze zou worden geïmplementeerd, evenals de voordelen ervan...

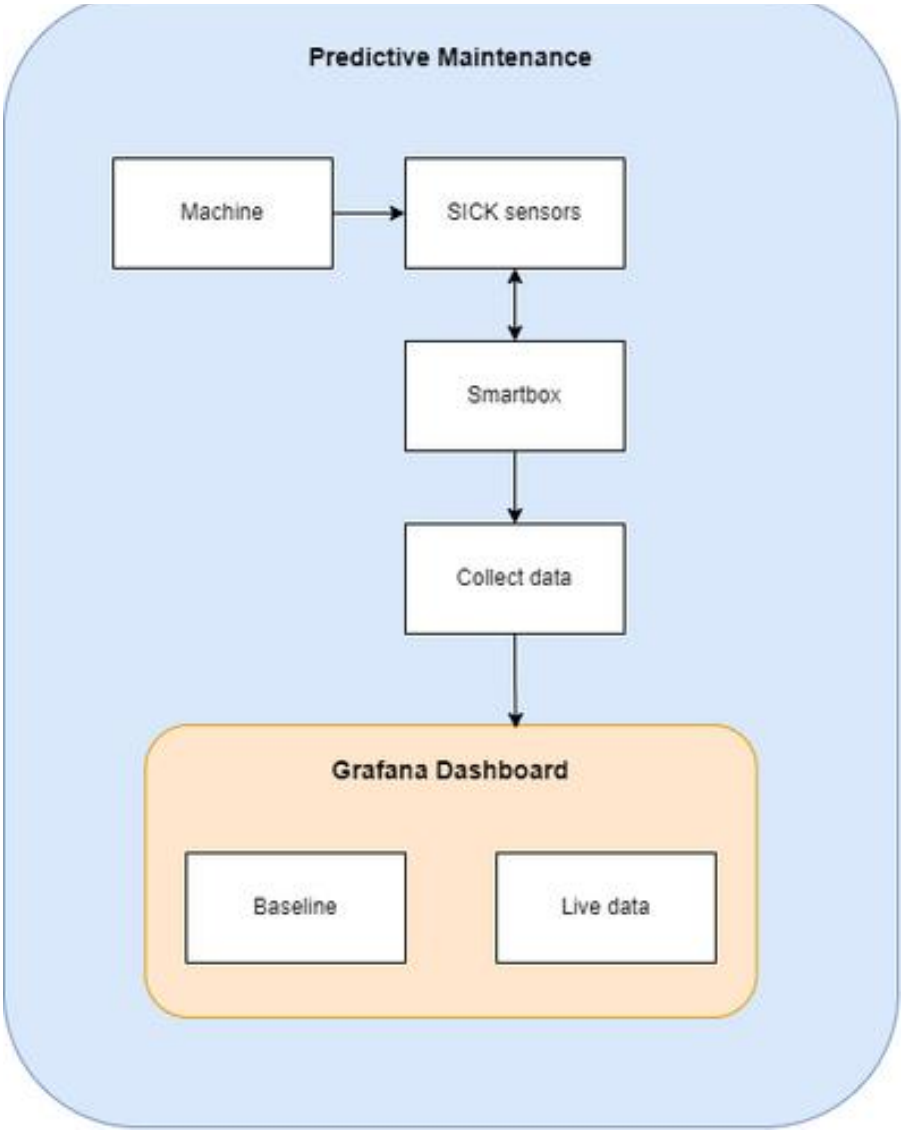
Wat is predictive maintenance?

Het is een proactieve benadering die gebruikmaakt van gegevens van machines om te voorspellen wanneer onderhoud moet worden uitgevoerd. Op deze manier kunnen we een probleem oplossen voordat het zich voordoet.

Wat zijn de voordelen van predictief onderhoud?

- Vermindert stilstand: minimaliseert de tijd dat machines niet operationeel zijn, waardoor de productie continu kan doorgaan.
- Voorkomt onverwachte uitval: identificeert potentiële problemen voordat ze leiden tot plotselinge uitval van apparatuur.
- Verlengt de levensduur van apparatuur: regelmatige monitoring en tijdig onderhoud houden machines in betere conditie, waardoor hun bruikbare levensduur wordt verlengd.
- Minder kosten: vermindert dure reparaties en algemene onderhoudsuitgaven.
- Optimaliseert onderhoudsschema's: maakt geplande onderhoudsactiviteiten mogelijk op basis van de werkelijke machinecondities, waardoor de efficiëntie en de toewijzing van middelen worden verbeterd.

Onze oplossing?



Figuur 33: Schema Fabriek Voorstelling

Op een machine kunnen we SICK-trillingsensoren installeren. Om de gegevens van deze sensoren te verzamelen, gebruiken we een smartbox. Op de smartbox kunnen we een nieuwe "functie" installeren om de gegevens van de SICK-sensoren te verzamelen. De SICK-sensoren blijven gegevens naar de smartbox sturen en de smartbox blijft deze verzamelen.
Deze gegevens kunnen worden gevisualiseerd in een Grafana-dashboard dat op elk scherm kan worden weergegeven. In dit dashboard vindt u informatie over de trillingsniveaus van de machines.
Het idee is dat aan de ene kant van het scherm een baseline staat. Deze baseline vertegenwoordigt hoe de trillingen van de machine zijn wanneer deze normaal functioneert. Aan de andere kant heeft u de live grafieken die elke X seconden worden bijgewerkt. U kunt ze dan vergelijken en verschillen/anomalieën opsporen.

Waarom trillingen?

Laten we de vergelijking maken tussen mens/machine.
Een mens heeft een hart met een hartslag. Deze hartslag kan worden gevisualiseerd met grafieken. In deze grafieken kunnen we allerlei informatie vinden: heeft hij een hoge/lage hartslag, onregelmatigheden in het hartritme, bloeddruk...
De machine trillingen zijn vergelijkbaar met het hartslag van een mens. Deze trillingen vertellen ons hoe de machine het doet. We kunnen ook grafieken maken van deze trillingen en ze analyseren. Het is mogelijk om trends, verschillen, anomalieën... in deze grafieken op te sporen.

Waarom een baseline?

Een baseline is nodig omdat we moeten weten hoe de machine normaal functioneert. We kunnen dan vergelijken met deze baseline om verschillen en mogelijke fouten op te sporen.

Welke gegevens worden gevisualiseerd in Grafana? Welke soorten grafieken hebben we nodig?

• Gemiddelde trillingen: geeft een algemeen overzicht van de trillingen van de machines. Het maakt het gemakkelijk om bepaalde trends op te sporen.
• Piektrillingen: visualiseert de pieken die worden gegenereerd door een machine (misschien door een component die elke 10 minuten schakelt).
• Heatmaps (van de gemiddelde trillingen): een meer visuele manier om onze gegevens weer te geven, om kleurenschema's te vergelijken.

Waarom zouden we het moeten gebruiken?

• Predictief onderhoud heeft vele voordelen zoals hierboven vermeld.
• Onze oplossing is eenvoudig te installeren omdat we al smartboxen gebruiken.
• Grafieken kunnen op elk scherm worden weergegeven om vergelijkingen te maken.
• U krijgt een beter beeld van hoe de machine presteert.

18. Bronnenlijst

1. SICK AG. Operating instructions sensor integration gateway SIG200 Ethernet [PDF]. Retrieved from https://cdn.sick.com/media/docs/7/87/887/operating_instructions_sensor_integration_gateway_sig200_ethernet_iptm_r_en_im0090887.pdf
2. SICK AG. Operating instructions MPB10 multi physics box [PDF]. Retrieved from https://cdn.sick.com/media/docs/3/03/403/operating_instructions_mpb10_multi_physics_box_en_im0102403.pdf
3. SICK AG. Operating instructions SIG200 REST API [PDF]. Retrieved from https://www.sick.com/media/docs/4/24/724/operating_instructions_sig200_rest_api_en_im0084724.pdf
4. Farnell. Using vibration analysis in predictive maintenance. Retrieved from https://uk.farnell.com/using-vibration-analysis-in-predictive-maintenance-trc-ar#:~:text=Vibration%20analysis%20allows%20engineers%20to,in%20equipment%20that%20use%20motors.
5. Advanced Technology. What is vibration analysis in predictive maintenance?. Retrieved from https://www.advancedtech.com/blog/what-is-vibration-analysis-in-predictive-maintenance/
6. IBM. (2023). Vibration analysis. Retrieved from https://www.ibm.com/blog/vibration-analysis/
7. Tractian. Vibration sensor predictive maintenance. Retrieved from https://tractian.com/en/blog/vibration-sensor-predictive-maintenance
8. Predictive maintenance with vibration analysis. (2023). Engineering Proceedings, 4(3), 102. https://www.mdpi.com/2673-4117/4/3/102
9. MaintainX. (2022). Using vibration analysis in predictive maintenance. Retrieved from https://www.getmaintainx.com/learning-center/using-vibration-analysis-in-predictive-maintenance
10. MathWorks. Predictive maintenance with MATLAB. Retrieved from https://nl.mathworks.com/campaigns/offers/predictive-maintenance-with-matlab.html?gclid=EAIaIQobChMIkpr48vnQhgMVpz0GAB3GcgHIEAAYAiAAEgL0EvD_BwE&ef_id=EAIaIQobChMIkpr48vnQhgMVpz0GAB3GcgHIEAAYAiAAEgL0EvD_BwE:G:s&s_kwcid=AL!8664!3!615072205573!e!!g!!predictive%20maintenance&s_eid=psn_68514676699&q=predictive+maintenance&gad_source=1
11. Fiix Software. Predictive maintenance strategies. Retrieved from https://fiixsoftware.com/maintenance-strategies/predictive-maintenance/
12. Grafana Labs. Grafana documentation. Retrieved from https://grafana.com/docs/grafana/latest/
13. Node-RED. Node-RED documentation. Retrieved from https://nodered.org/docs/

14. Elastic. Elasticsearch documentation. Retrieved from https://www.elastic.co/guide/index.html?utm_campaign=Google-B-EMEA-N-Exact&utm_content=Brand-Core-Documentation&utm_source=google&utm_medium=cpc&device=c&utm_term=elasticsearch%20docs&gad_source=1&gclid=EAIaIQobChMI_9f2lPrQhgMV5UpBAh3Y1gZjEAAYASAAEgInJfD_BwE
15. Red Hat, Inc. Ansible documentation. Retrieved from https://docs.ansible.com/
16. ScienceDirect. (2023). Noise and vibration analysis for improved product quality. Retrieved from https://www.sciencedirect.com/science/article/pii/S2046043023000308
17. Saab RDS. Noise and vibration analysis for improved product quality. Retrieved from https://saabrds.com/noise-and-vibration-analysis-for-improved-product-quality/
18. OpenAI. ChatGPT. Retrieved from https://chatgpt.com/