

TECHNICAL REPORT– DRAFT VERSIE

**Naam student:** Arda Karakaya  
 **Studentnummer:** 526942  
 **Bedrijf:** Strukton Systems BV  
 **Afdeling:** Software Engineering – Hengelo  
 **Datum:** 25 oktober 2025  
 **Versie:** 0.1

## Foreword

Dit verslag presenteert de draftversie van de technische opdracht die tijdens de stage bij Strukton Systems BV is uitgevoerd. Het doel van het project is het ontwikkelen van een prototype voor het verwerken van sensordata op het rangeerterrein in Zutphen.

De stage bood de mogelijkheid om praktische ervaring op te doen op het gebied van softwareontwikkeling, databasebeheer en systeemintegratie. Het werk dat in dit verslag wordt beschreven, is uitgevoerd onder begeleiding van zowel de bedrijfsbegeleider als de Saxion-begeleider.

## Summary

The purpose of this project was to develop a prototype system for processing sensor data at the shunting yard in Zutphen, operated by Strukton Systems BV. Traditional railway infrastructure uses wheel counters and track circuits to determine track section occupancy, which are highly reliable but complex and costly. However, not all track sections are monitored, such as shunting yards and storage tracks. To provide insight into these non-monitored areas, Strukton developed a battery-powered IoT wheel passage sensor, capable of recording wagon movements and transmitting the data wirelessly. The main goal of the project was to design a system that processes this sensor data and gives a insight into what happens on the tracks.

The project followed the Waterfall methodology, structuring the work into different phases of analysis, design, realisation, testing, and delivery. The analysis included gathering requirements and studying sensor placement on the yard. During the design phase, the layout, sensor placement and database connections were prepared. The realisation phase involved setting up the software in C# and a SQL Server database, with version control maintained in GitHub. Preliminary tests were conducted to verify correct data collection and storage.

The results so far show that the system is functional and can collect, store, and process sensor data. However, delays have been observed in the switch position sensors, which prevent accurate determination of wagon direction.

Major conclusions include the effectiveness of a structured Waterfall approach for project planning and execution. Further testing and analysis are needed to address the observed sensor delays and to validate the system under operational conditions.

Inhoud

[Foreword 2](#_Toc104398307)

[Summary 3](#_Toc514492131)

[Abbreviations 5](#_Toc1379152358)

[List of Figures and Tables 6](#_Toc1421143079)

[1 Introduction 7](#_Toc2087931797)

[1.1 Background of the Assignment 8](#_Toc696235267)

[1.3 Scope of Work 8](#_Toc1647340032)

[1.4 Approach & Methodology 9](#_Toc1310346957)

[1.5 Outline of the Report 9](#_Toc374518741)

[2 Analyse / specifications. 10](#_Toc1423326817)

[2.1 Stakeholders 11](#_Toc2038032814)

[2.2 System Context 11](#_Toc603040626)

[2.3 Requirements 11](#_Toc1687057556)

[2.3.1 Functional Requirements (FR) 11](#_Toc1871870568)

[2.3.2 Non-Functional Requirements (NFR) 12](#_Toc640534404)

[2.4 Risk Analysis 12](#_Toc518341244)

[3 Functional design 14](#_Toc1025757226)

[3.1 Process Overview 15](#_Toc1880399007)

[3.2 Functionele Componenten 15](#_Toc638846920)

[3.3 User Stories 16](#_Toc487546856)

[3.4 Explanation of Functionalities 17](#_Toc844280073)

[4 Technical design 17](#_Toc1871519949)

[4.1 Software Architectuur 18](#_Toc1554955915)

[4.2 Direction & Assentelling Logic 18](#_Toc1905366292)

[4.3 Software Componenten 20](#_Toc1217541017)

[5 Realisation 20](#_Toc1362071164)

[5.1 Project setup 21](#_Toc2017450810)

[5.2 Implemented Features 21](#_Toc1204091074)

[5.3 Challenges and Solutions 21](#_Toc319016854)

[5.4 Summary of Implementation 22](#_Toc776904758)

[6 Testing 22](#_Toc515546183)

[6.1 Test Strategy 23](#_Toc1880237748)

[6.2 General Test Results 23](#_Toc1296254827)

[6.3 Specific Test Case: Unexpected Negative Values in Exce 23](#_Toc150051884)

[6.3.1 Findings 23](#_Toc1533697316)

[6.3.1.1 Expected Data Flow According to Switch Position 24](#_Toc1494059902)

[6.3.1.2 Actual Data Flow 24](#_Toc867844178)

[6.3.1.3 Consequence: Incorrect Movement Reconstruction → Negative Values 24](#_Toc455136380)

[6.3.1.4 Possible Cause: Delay in Switch Position Sensors 25](#_Toc408597370)

[6.3.1.5 Test Result: Confirmed Delay in Switch Position Registration 25](#_Toc353145922)

[7 Conclusions and Recommendations 26](#_Toc913301903)

[7.1 Conclusions 27](#_Toc1183367477)

[7.2 Recommendations 27](#_Toc1526825348)

[References 27](#_Toc302065966)

[Appendices 28](#_Toc1957401220)

## Abbreviations

|  |  |
| --- | --- |
| **Abbreviation** | **Meaning** |
| WPS | Wheel Passage Sensor |
| IoT | Internet of Things |
| SQL | Structured Query Language |
| API | Application Programming Interface |
| UI | User Interface |

## List of Figures and Tables

## 1 Introduction

### 1.1 Background of the Assignment

Strukton Systems BV ontwikkelt en onderhoudt besturings- en monitoringssystemen voor spoorweginfrastructuur. Binnen de afdeling Software Engineering zijn projecten gericht op het verbeteren van de efficiëntie en veiligheid op spoorlijnen. Dit project richt zich op het verwerken van sensordata van het rangeerterrein in Zutphen. Het doel is om assen op elk spoorsectie automatisch te detecteren en deze informatie te verwerken, zodat de bezetting en verkeersdrukte van de sporen inzichtelijk worden gemaakt en het operationele overzicht wordt verbeterd.

1.2 Problem Definition and Objectives

Assentellers en spoorstroomlopen worden al jarenlang gebruikt om de bezetting van spoorsecties vast te stellen. Deze systemen zijn betrouwbaar, maar complex en kostbaar, waardoor niet elk spoor hiermee wordt bewaakt. Dit geldt bijvoorbeeld voor rangeerterreinen en opstelplaatsen.

Om inzicht te krijgen in deze zogenaamde niet-beheerde gebieden, heeft Strukton een batterijgevoede IoT-wielpassagesensor ontwikkeld. Deze sensor registreert bewegingen van wagons en ontsluit de data draadloos.

Het probleem en de aanleiding van deze opdracht is dat er momenteel geen gestructureerd systeem bestaat om deze sensorinformatie te verzamelen, verwerken en opslaan. Het doel van het project is daarom het ontwikkelen van een prototype dat sensordata kan verzamelen, verwerken, zodat Strukton inzicht krijgt in de bezetting en bewegingen op het rangeerterrein.

### 1.3 Scope of Work

**In scope:**

* Het verzamelen en verwerken van sensordata van het rangeerterrein.
* Ontwikkeling van software in C# voor het inlezen, verwerken en opslaan van data.
* Uitvoeren van voorlopige tests van de functionaliteit.

**Out of scope:**

* Integratie met live operationele systemen.
* Ontwikkeling van een grafische gebruikersinterface of visualisatieplatform.

### 1.4 Approach & Methodology

Het project volgt de Waterfall-methodologie, waarbij het werk wordt verdeeld in opeenvolgende fasen: analyse, ontwerp, realisatie, testen en oplevering. Deze aanpak biedt structuur en overzicht tijdens de uitvoering van een individueel stageproject.

* **Analysefase:** eisen en specificaties verzameld via gesprekken met de bedrijfsbegeleider en teamleden.
* **Ontwerpfase:** bestudering van het rangeerterrein, bepalen van sensorlocaties en as-tellingen.
* **Realisatiefase:** de software wordt ontwikkeld in C# en de sensordata opgeslagen in een SQL Server database. GitHub wordt gebruikt voor versiebeheer en voortgangsbewaking.
* **Testfase:** voorlopige tests uitgevoerd om te controleren of sensordata correct wordt verwerkt.
* **Opleveringsfase:** vindt plaats aan het einde van de stageperiode, inclusief werkend prototype en documentatie.

### 1.5 Outline of the Report

Het verslag bestaat uit de volgende hoofdstukken:

* **Introduction:** achtergrond, probleemstelling, doelstelling, scope, methodologie en opbouw van het verslag.
* **Analyse / Specifications:** systeemvereisten, sensor- en databasedetails.
* **Functional Design:** beschrijving van functies en processen van het systeem.
* **Technical Design:** technische componenten, database-ontwerp en interfaces.
* **Realisatie:** opzet van de software en database, implementatie.
* **Testing:** uitgevoerde tests en voorlopige resultaten.
* **Conclusions and Recommendations:** conclusies en aanbevelingen voor vervolg.

## 2 Analyse / specifications.

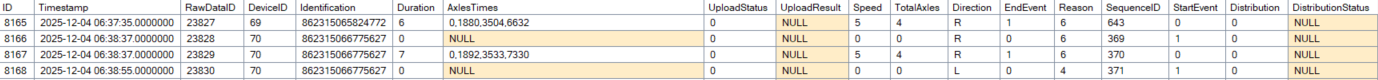
In deze fase zijn alle eisen, randvoorwaarden en systeemkarakteristieken onderzocht.

### 2.1 Stakeholders

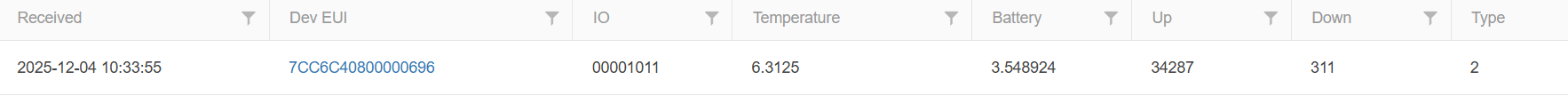
• Software Engineering Team Strukton  
 • Beheerders rangeerterrein Zutphen  
 • Saxion Hogeschool (begeleiding)

### 2.2 System Context

De WPS- en wisselstand sensoren sturen data naar een interne database. Het prototype verwerkt deze gegevens en slaat deze op in een SQL-database.

**Informatie die per WPS sensor beschikbaar is:**

* Tijdstip wielpassage
* Tijdstip waarin assen onderling over de sensoren gaan
* Sensor ID
* Snelheid
* Richting van de wagon

**Informatie die per wisselstand sensor beschikbaar is:**

* Richting van de wissel
* Device EUI
* Temperatuur
* Batterij
* Type; waarbij type 2 een update bericht is en type 4 een verandering bericht is

### 2.3 Requirements

#### 2.3.1 Functional Requirements (FR)

* **FR1 – Data-ophalen**  
  Het systeem moet automatisch nieuwe sensordata (WPS- en wisselstandsensoren) kunnen ophalen uit de SQL-database.
* **FR2 – Dataverwerking**  
  Het systeem moet de opgehaalde ruwe sensordata kunnen verwerken, valideren en omzetten naar data die gebruikt kan worden.
* **FR3 – Richting-bepaling spoorbeweging**  
  Het systeem moet kunnen bepalen of een as een spoor op rijdt of af rijdt, op basis van WPS data.
* **FR4 – Richting-bepaling door wisselstandsensoren**  
  Het systeem moet rijrichting kunnen bepalen door gebruik te maken van wisselstand-sensordata.
* **FR5 – Assentelling tussen wissels**  
  Het systeem moet het aantal assen kunnen tellen dat tussen twee wisselstanden (van wissel A naar wissel B) over het terrein heeft gereden.
* **FR6 – Sensorhistorie (WPS)**  
  Het systeem moet een event handler beschikbaar hebben die per WPS-sensor de volledige geschiedenis van gedetecteerde events inzichtelijk maakt.
* **FR7 – Sensorhistorie (wisselstandsensoren)**  
  Het systeem moet een event handler beschikbaar hebben die per wisselstandsensor laat zien welke standen en events zich in de tijd hebben voorgedaan.
* **FR8 – Exportfuncti**onaliteit  
  Het systeem moet de verwerkte resultaten kunnen exporteren naar een Excel-bestand.

#### 2.3.2 Non-Functional Requirements (NFR)

* **NFR1**: Betrouwbaarheid — gegevens moeten correct worden verwerkt
* **NFR2**: Prestatie — binnen enkele seconden verwerken
* **NFR3**: Beheerbaarheid— code moet gestructureerd en leesbaar zijn
* **NFR4**: Veiligheid — data blijft binnen Strukton-infrastructuur

### 2.4 Risk Analysis

Tijdens het project zijn risico’s geïdentificeerd die de betrouwbaarheid van de resultaten en de tijdige oplevering kunnen beïnvloeden. Hieronder staan de belangrijkste risico’s, inclusief de impact en mitigerende maatregelen.

**R1 – Sensorvertraging en inconsistente timing**

* **Beschrijving:**  
  De wisselstandsensoren verzenden soms vertraagd of onregelmatig berichten door. Deze vertraging zorgt ervoor dat de geregistreerde tijdstempels niet altijd overeenkomen met de werkelijke volgorde van gebeurtenissen.
* **Effect:**  
   • Onjuiste richtingberekeningen  
   • Verkeerde telling van assen tussen wissels  
   • Onbetrouwbare passagegroepering
* **Mitigatie:**  
   • Tijdstoleranties toepassen (bijv. ±1–3 seconden)  
   • Meerdere berichten verzenden

**R2 – Complexiteit van de dataverwerking en logica**

* **Beschrijving:**  
  De logica voor passagegroepering, wisselinterpretatie en richting-bepaling is complex. Kleine fouten in berekeningen of in de interpretatie van tijdstempels kunnen grote invloed hebben op de eindresultaten.
* **Effect:**  
   • Onjuiste algoritme-uitvoer  
   • Moeilijk debuggen van dataproblemen  
   • Vertraging in het ontwikkelproces
* **Mitigatie:**  
   • De logica opdelen in kleine, testbare modules  
   • Unit tests uitvoeren op elke berekening  
   • Logging toevoegen voor elke stap in de dataverwerking

**R3 – Afhankelijkheid van externe systemen en begeleiders**

* **Beschrijving:**  
  Het prototype is afhankelijk van de centrale SQL-database van Strukton. Wanneer deze systemen tijdelijk niet beschikbaar zijn, vertraagt dat het ontwikkelproces.
* **Effect:**  
   • Tijdelijke onderbreking in dataverwerking  
   • Vertraging in testen en validatie  
   • Mogelijke uitloop van de planning
* **Mitigatie:**  
   • Dummy data gebruiken bij storingen  
   • Regelmatige afstemming met begeleiders plannen

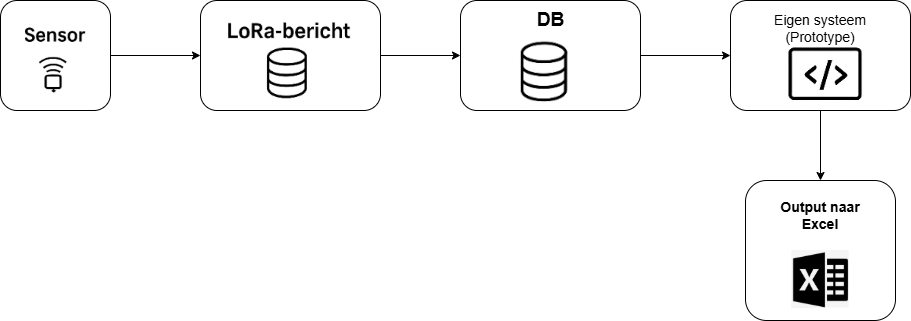
## 3 Functional design

Het functioneel ontwerp beschrijft wat het systeem moet doen om de gestelde functionele eisen te realiseren. Het geeft inzicht in de processen, componenten en gebruikersinteracties van het prototype.

### 3.1 Process Overview

Het prototype verwerkt sensordata van WPS- en wisselstandsensoren volgens het volgende proces:

1. **Sensorsturing**
   * WPS- en wisselstandsensoren sturen berichten met passage- en wisseldata.
2. **Dataopslag in centrale database**
   * Binnen Strukton wordt de ruwe sensorinformatie opgeslagen in een centrale SQL-database.
3. **Dataophalen door prototype**
   * Het prototype haalt nieuwe data op via directe SQL-connecties.
4. **Dataverwerking volgens logica**
   * Sensorberichten worden verwerkt, foutieve waarden worden gefilterd en passages worden gegroepeerd.
   * De rijrichting wordt berekend op basis van WPS- en wisselstandsensoren.
5. **Output naar Excel**
   * Het verwerkte resultaat wordt verzonden naar Excel.



### 3.2 Functionele Componenten

Het systeem bestaat uit vijf hoofdcomponenten die samenwerken om sensordata te verwerken en bruikbare informatie te genereren:

**1. Data Module**

* Leest nieuwe sensor waardes in vanuit de database.
* Zorgt dat het prototype continu toegang heeft tot actuele sensordata.

**2. Pre-Processor**

* Bereidt de opgehaalde data voor op verdere verwerking.
* Verwerkt de ruwe sensordata zodat deze gebruikt kan worden voor passagegroepering en richting-bepaling.

**3. Event Module**

* Groepeert individuele sensorberichten tot overzichtelijke passage-events.
* Creëert event handlers die per passage de volledige sensorhistorie weergeven.
* Deze event handlers bevatten al verwerkte en bruikbare informatie, zoals:
  + Tijdsvolgorde van passages
  + Aantal assen dat is gepasseerd per WPS sensor met richting
  + Rijrichting per wisselstand sensor met tijd
  + Wisselstanden tijdens de passage
* Hierdoor kan een gebruiker direct analyseren wat er tijdens een passage is gebeurd, zonder dat extra verwerking nodig is.

**4. Richtings- en Assentelling Logica**

* Berekent de rijrichting van wagons, zowel op als af het terrein.
* Bepaalt het aantal assen dat tussen twee wisselstanden passeert.
* Gebruikt sensordata van meerdere sensoren om een betrouwbare bepaling te maken.

**5. Output / Database Writer**

* Slaat de verwerkte gegevens en passage-events op van het prototype.
* Maakt export naar Excel files.

### 3.3 User Stories

Het systeem ondersteunt de volgende belangrijke gebruikersbehoeften:

|  |  |  |
| --- | --- | --- |
| **Gebruiker** | **Behoefte** | **Doel / Waarde** |
| **Ingenieur / Data-analist** | Wielpassages overzichtelijk kunnen uitlezen | Snel analyseren hoeveel assen tussen wissels zijn gereden en bewegingen reconstrueren |
| **Rangeerterreinoperator** | Inzicht in passage-events en rijrichting van wagons op het terrein | Efficiënter toezicht houden en mogelijke knelpunten op het terrein herkennen |
| **Projectbegeleider / Supervisor** | Resultaten en overzichten van sensordata kunnen bekijken en interpreteren | Voortgang monitoren, beslissingen onderbouwen en prototype evalueren |

### 3.4 Explanation of Functionalities

Het systeem verwerkt de sensordata van de WPS- en wisselstandsensoren en zet deze om in bruikbare informatie voor analyse. Het systeem kan onder andere:

* Passages van wagons groeperen en inzicht geven per sensor
* De rijrichting van wagons bepalen, zowel op als van het terrein
* Het aantal assen dat tussen wissels passeert berekenen
* Event handlers maken die per passage de volledige geschiedenis laten zien
* Resultaten exporteren naar Excel voor verdere analyse

Dankzij de modulaire opbouw kan het systeem eenvoudig uitgebreid worden met nieuwe sensoren of extra functies in de toekomst. De ontwikkeling is gestructureerd uitgevoerd: eerst wordt data opgehaald en voorbereid, daarna worden passages gegroepeerd en de richting bepaald, en ten slotte worden de resultaten opgeslagen en beschikbaar gemaakt voor analyse.

## 4 Technical design

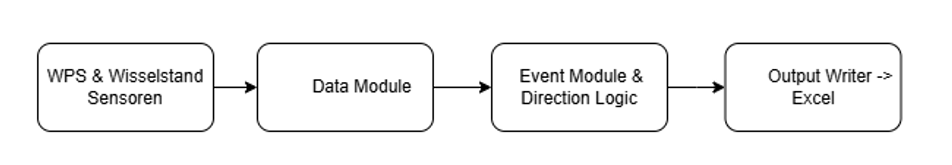
Dit hoofdstuk beschrijft de technische opzet van het prototype, de softwaremodules, het verwerkingsproces van sensordata en de richting- en assentellinglogica.

### 4.1 Software Architectuur

Het systeem is modulair opgebouwd en bestaat uit drie hoofdcomponenten:

1. **Data Module**
   1. Haalt sensordata op van de WPS- en wisselstandsensoren.
   2. Zorgt dat alle nieuwe berichten beschikbaar zijn voor verwerking.
2. **Event Module & Direction Logic**
   1. Groepeert individuele sensorberichten tot passage-events.
   2. Event handlers tonen per passage de verwerkte, bruikbare informatie, zoals:
      * Tijd van passages
      * Aantal assen per WPS sensor
      * Rijrichting per wisselstandsensor
      * Wisselstanden tijdens de passage
   3. Berekening van aantal assen:
      * Tijd tussen passages van sensoren
      * Richting van wisselstand
      * Assen per wissel tot wissel
3. **Output Writer**

* Verwerkt resultaten worden direct geëxporteerd naar Excel.
* Maakt de analyse van passage-events en sensorinformatie mogelijk zonder database.



### 4.2 Direction & Assentelling Logic

De richting van de wagons en het aantal assen worden bepaald op basis van:

* IO-waardes van de wisselstand sensoren 
* TotalAxles-waardes van de WPS sensoren 
* Direction-waardes van de WPS sensoren 

Door deze sensorgegevens te combineren met de bekende locaties van de sensoren op het spoor, is een algoritme ontwikkeld dat kan bepalen hoeveel assen zich op elk deel van het rangeerterrein bevinden.

**Werkwijze:**

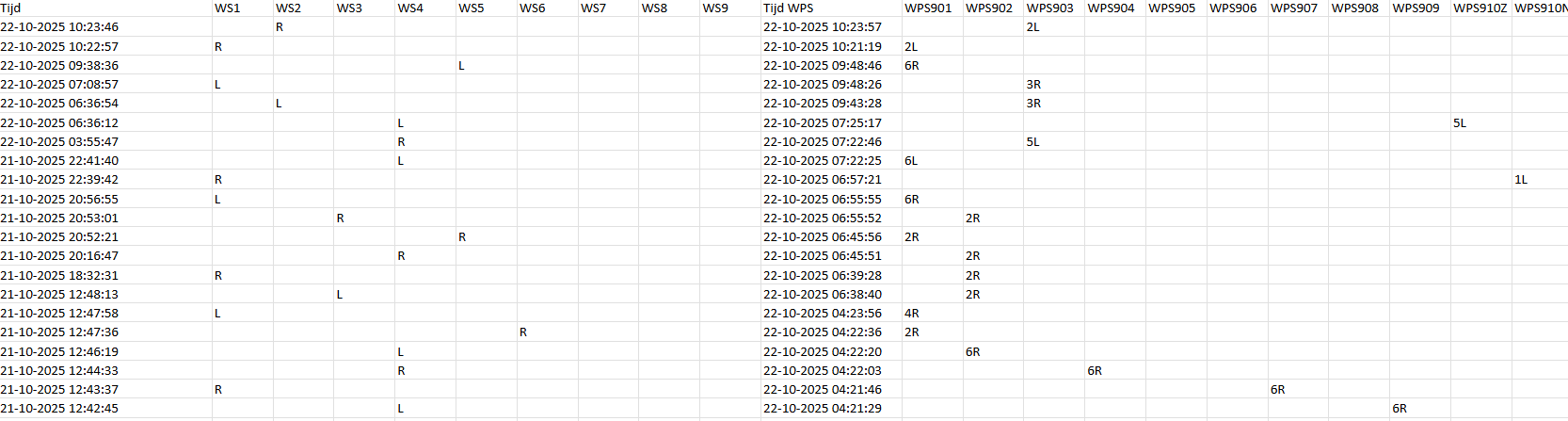
1. De tijdstippen en richtingen van de wisselstandsensor-berichten worden eerst gegroepeerd.
2. Voor elk WPS-sensorbericht wordt bepaald welke sensor het bericht heeft verzonden.
3. De tijd van het WPS-bericht wordt vergeleken met de richting van de bijbehorende wisselstandsensor.
4. Vervolgens worden de assen van de WPS-sensor gegroepeerd op basis van passage en tijd.
5. De rijrichting van de wagen wordt afgeleid (op of van het terrein).
6. Afhankelijk van de combinatie van WPS-sensorrichting en wisselstandsensorrichting worden assen verwijderd van eerdere spoorsegmenten.
7. Afhankelijk van de combinatie van WPS-sensorrichting en wisselstandsensorrichting worden het aantal assen dat daadwerkelijk passeert opgeteld en gekoppeld aan de juiste spoorsegment.

**Beperkingen en oplossingen**

* **Beperkingen:** Wisselsensoren kunnen vertraagde signalen doorgeven, waardoor de bepaling van de rijrichting tijdelijk onzeker kan zijn.
* **Oplossing:** Het systeem past tijdtoleranties toe.

**Event handlers**

* Per sensor tonen de event handlers direct de verwerkte en bruikbare informatie.
* Gebruikers kunnen hierdoor zonder extra berekeningen of verwerking analyseren wat er op een bepaald tijdstip op een sensor is geregistreerd.



### 4.3 Software Componenten

* **C#** – Programma die gebruikt wordt om de data te verwerken.
* **GitHub** – versiebeheer en samenwerking tijdens ontwikkeling.
* **Logging** – registreert belangrijke stappen en fouten in de verwerking, zodat problemen traceerbaar zijn.

**Toelichting keuzes:**

* C# biedt stabiliteit en snelle ontwikkeling voor modulair opgebouwde applicaties en wordt tevens binnen Strukton altijd gebruikt voor back-end ontwikkeling.
* Modulaire opbouw maakt het systeem onderhoudbaar en uitbreid baar.

## 5 Realisation

In dit hoofdstuk wordt de implementatie van het prototype beschreven, inclusief de projectopzet, uitgevoerde functionaliteiten en de uitdagingen die tijdens de realisatie zijn ondervonden.

### 5.1 Project setup

* Het project is opgezet in Visual Studio als een C#.
* Versiebeheer is ingericht via GitHub, waarmee de ontwikkeling, versiecontrole en samenwerking overzichtelijk wordt beheerd.
* Het prototype maakt verbinding met de sensoren via de centrale database, zodat real-time en historische data beschikbaar is voor verwerking.
* Logging is geïmplementeerd om belangrijke stappen in het verwerkingsproces en eventuele fouten vast te leggen, wat traceerbaarheid en foutopsporing vergemakkelijkt.

### 5.2 Implemented Features

Het prototype bevat de volgende kernfunctionaliteiten:

1. Data ophalen – Nieuwe sensordata van WPS- en wisselstandsensoren wordt automatisch ingelezen.
2. Data voorbereiding en verwerking – Sensorwaarden worden gecontroleerd en verwerkt tot bruikbare gegevens voor passagegroepering en richting-bepaling.
3. Passagegroepering – Individuele sensorberichten worden samengevoegd tot overzichtelijke passage-events, waarbij event handlers per passage de volledige sensorhistorie tonen.
4. Richting- en assentelling – Het systeem bekijkt de rijrichting van wagons (op of van het terrein), richting van de wissels en telt het aantal assen dat per passage over het terrein rijdt.
5. Output / Excel-export – Verwerkte resultaten worden direct geëxporteerd naar Excel, zodat gebruikers eenvoudig analyses kunnen uitvoeren zonder extra verwerking of een database nodig te hebben.

### 5.3 Challenges and Solutions

Tijdens de realisatie van het prototype zijn enkele uitdagingen opgetreden:

* **Onregelmatige sensordata**  
   Wisselstand sensoren zenden vertraagd berichten, wat de verwerking kan bemoeilijken.  
  **Oplossing:** Het systeem past tijdtoleranties toe en er worden inmiddels door Strukton vier berichten verzonden wat de kans op kleinere vertragingen vergroot.

### 5.4 Summary of Implementation

Het resultaat is een stabiel, modulair en uitbreid baar prototype dat sensordata kan ophalen, verwerken, groeperen, richting bepalen en het aantal assen berekenen. De verwerkte resultaten worden direct beschikbaar gesteld in Excel, waardoor gebruikers eenvoudig inzicht krijgen in passage-events en wagonbewegingen op het rangeerterrein.

Het prototype vormt een solide basis voor verdere ontwikkeling, bijvoorbeeld het toevoegen van nieuwe sensoren, extra analysemogelijkheden of integratie met operationele systemen of met wisselstand sensoren die geen vertraging hebben.

## 6 Testing

### 6.1 Test Strategy

Het testen van het prototype is uitgevoerd op drie niveaus:

* **Unit tests:**  
  Testen van de logica voor passagegroepering, richting-bepaling en assentelling.
* **Integratietests:**  
  Verifiëren dat de koppeling tussen sensordata (WPS en wisselstand) en de verwerking correct werkt.
* **Handmatige tests:**  
  Controle van individuele passages en assenregistraties op juistheid en consistentie met real-time waarnemingen.

### 6.2 General Test Results

* Data wordt correct opgehaald en verwerkt.
* Passagegroepering logica werkt zoals verwacht.
* Richting-bepaling is grotendeels correct, afhankelijk van de tijdigheid van de sensor berichten.
* Event handlers tonen de verwerkte en bruikbare informatie zoals bedoeld.
* Foutmeldingen worden correct geregistreerd en kunnen door de gebruiker worden ingezien.

### 6.3 Specific Test Case: Unexpected Negative Values in Exce

**Aanleiding:**  
Tijdens de export naar Excel werden negatieve waarden aangetroffen, wat duidt op een probleem bij koppeling van assen aan spoorsegmenten.

**Testmethode:**

* Data vergeleken met real-time en historische logs
* Sensorvolgorde gevalideerd
* Assenregistraties handmatig nagelopen
* Wisselconfiguraties gecontroleerd op timestamp-niveau

##### 6.3.1 Findings

**Sensor bij wissel 906 ontvangt 12 assen**



Uit de database is vastgesteld dat de sensor bij wissel 906 correct 12 assen registreerde.



Op basis van de wisselconfiguratie had deze treinbeweging via de rechterstand naar spoor B moeten gaan (zie configuratieschema). In de configuratie wordt dit bepaald door de vijfde bit van links: een waarde van 0 betekent dat de wissel rechts staat, een waarde van 1 betekent dat de wissel naar links staat. Op het relevante tijdstip correspondeerde IO op 135 wat dus de vijfde bit een 0 maakt, en dit geeft aand dat de wissel op rechts stond.

##### 6.3.1.1 Expected Data Flow According to Switch Position

**Wisselconfiguratie (verwacht):**  
Sensor bij wissel 906 → wissel stuurt verkeer naar rechts→ spoor B zou in dit scenario in totaal 12 assen moeten bijtellen.

##### 6.3.1.2 Actual Data Flow

Wanneer de daadwerkelijke assen worden gevolgd, komen de assen niet terecht bij de verwachte spoor B, maar bij de sensor die staat bij wissel 908.

Dit is afwijkend, omdat:

* De sensor van wissel 908 niet hoort bij de rechterstand, maar bij de linkerstand
* De wisselstand in de database aangaf dat de wissel rechts stond,
* De gesignaleerde route dus niet overeenkomt met de geregistreerde wisselstand.

##### 6.3.1.3 Consequence: Incorrect Movement Reconstruction → Negative Values

Omdat de code of Excel-sheet ervan uitging dat de wissel op rechts stond, werd de data gekoppeld aan de verkeerde sensorvolgorde.  
Hierdoor ontstond:

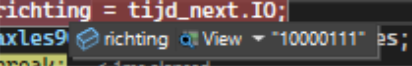
* een negatief verschil in aantal assen,
* een verkeerde reeks van berekeningen,
* en uiteindelijk negatieve waarden in Excel-berekeningen.

##### 6.3.1.4 Possible Cause: Delay in Switch Position Sensors

Tijdens de analyse is vastgesteld dat de timestamp van de wisselstand-bericht later geregistreerd werd dan verwacht.  
Dit wijst op een mogelijke delay tussen fysieke wisselstand en sensor bericht.

Dit zou betekenen:

* Dat de wissel fysiek links stond,
* Maar het systeem registreerde dit pas enkele seconden later,
* Waardoor de software de verkeerde logica toepaste.



(ad\fbeelding zie je dat bij die wps bericht de code de bijbehorende wisselstadn bericht check en dir\e idd rechts geeft)

Gevolg:  
De applicatie dacht dat de wissel rechts stond en stuurde het aantal assen naar de rechter deel van wissel 906 terwijl in werkelijkheid de trein naar links ging bij de wissel naar wissel 908.

##### 6.3.1.5 Test Result: Confirmed Delay in Switch Position Registration

Tijdens een test is gecontroleerd of de wisselstand op tijd de berichten verzond.  
De wissel werd:

* om 09:39 handmatig naar Rechts (R) gezet,
* om 09:40 handmatig naar Links (L) gezet.

In de database verschijnen echter de volgende logmeldingen (zie screenshot):



* 09:44:36 → wissel staat nog op de oude stand (Rechts)
* 09:52:59 → wissel registreert pas de nieuwe stand (Links)

Dit bevestigt een vertraging van 8 minuten tussen wissel verandering en het ontvangen van het bericht en bij de opeenvolgende tests waren de vertragingen ook vaak tussen 1,5 tot 2 minuten.

## 

## 7 Conclusions and Recommendations

### 7.1 Conclusions

Het uitgevoerde stageproject heeft geleid tot een werkend prototype dat voldoet aan de meeste gestelde functionele eisen. Belangrijke conclusies zijn:

* Het prototype kan sensordata van WPS- en wisselstandsensoren succesvol ophalen, verwerken en exporteren naar Excel.
* Passagegroepering en assentelling werken correct, waardoor inzicht wordt verkregen in het verkeer op het rangeerterrein.
* Er zijn twee duidelijke Event Handlers ontwikkeld die per WPS- en wisselstandsensor alle relevante informatie tonen: tijdstip, richting van de trein, richting van de wisselstand en aantal assen.
* De bepaling van de rijrichting van wagons is nog niet volledig betrouwbaar, vooral door vertragingen in de wisselstand-sensoren.
* De gestructureerde aanpak volgens de Waterfall-methodologie bood voldoende houvast en overzicht, waardoor de ontwikkeling van de verschillende modules systematisch kon plaatsvinden.

### 7.2 Recommendations

Om de functionaliteit en betrouwbaarheid van het systeem verder te verbeteren, worden de volgende aanbevelingen gedaan:

* **Verbetering/oplossing van sensorvertragingen:** vertraging oplossen of minimaal maken zouden de nauwkeurigheid van de assentelling en richting-bepaling verhogen.
* **Ontwikkeling van een visualisatietool:** een web- of app-gebaseerd dashboard kan het analyseren van passage-events en assentellingen vereenvoudigen en de bruikbaarheid voor operators vergroten.

## References

## Appendices