

IoT Rail Hackathon 2023

Team #1

Alexander Braml

Andreas Weber

Frederic Reiter

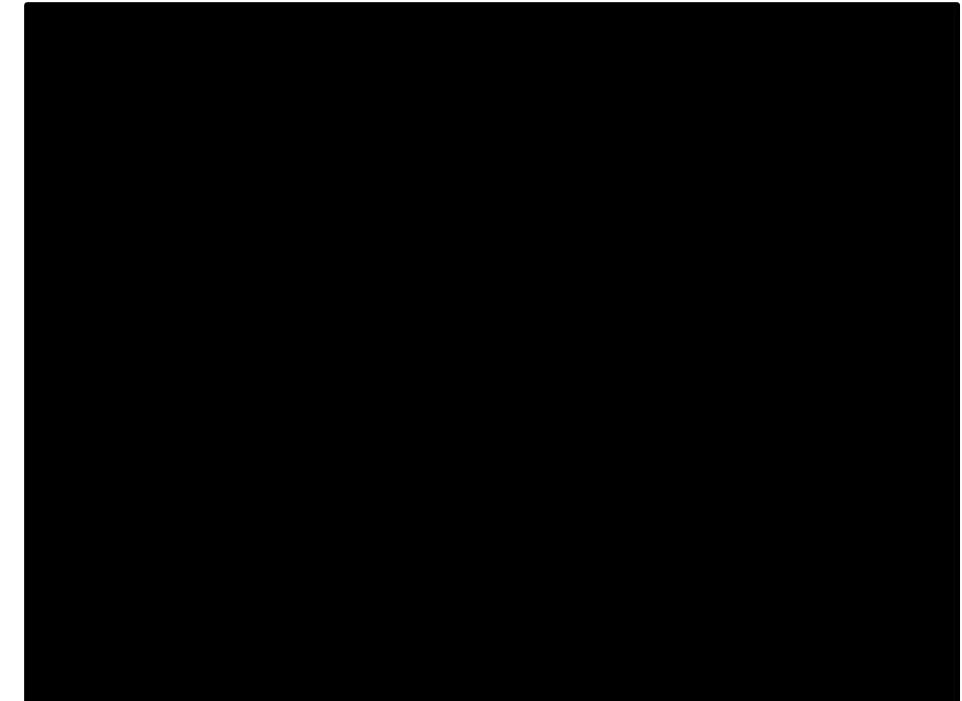
Lukas Rost

Niels Geist

Niklas Fischer

Peter Schroeder

- 3 Teilprojekte: Remote Diagnostics (Switch), Automatic Train Operation (Digital Signature), Remote Controlling (Axe Counting)
- Mögliche Lösungsansätze bereits skizziert
- Umsetzung anhand eines Modellbahn-Rings inkl. einer abschnittsweisen Parallelgleises als Ausweichgleis



Projektstrukturplanung

Remote Diagnostics: Switch



Frederic
TZFS



Peter
DB Systel, HWR Berlin

Remote Steering: Axle Counting



Andi
Universität Passau



Niklas
TU Chemnitz

Automatic Train Operations: Digital Signature



Lukas
Hasso-Plattner-Institut



Niels
TU Berlin

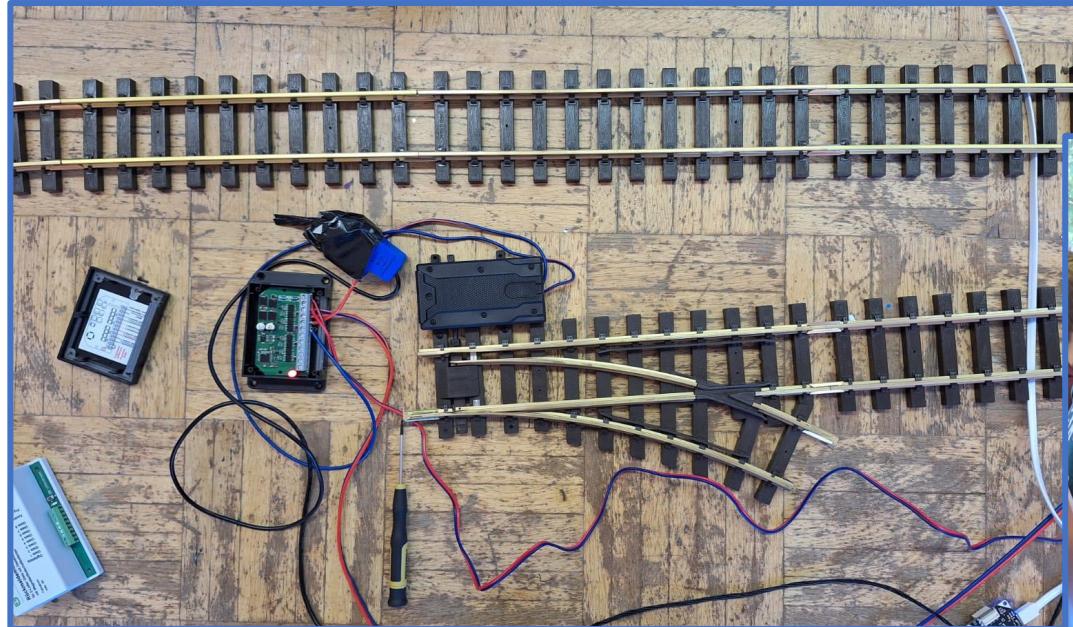


Alexander
Universität Passau

...durch die Aufteilung in Teilprojekte ist eine hochgradig parallelisierte und agile Arbeitsweise auf Augenhöhe gelungen

- Aufteilung in Teilprojekte
- Bearbeitung der Problemstellung in jeweiliger Teilgruppe, aber auch übergreifende Kollaboration
- sehr agiler Ansatz durch Zusammenarbeit auf Augenhöhe und kompetenzbasierte Aufgabenzuordnung ohne präzise Rollenverteilung
- hochgradig parallele Arbeitsweise

Prozessschritte - Remote Diagnostics



Aufbau:

- Verschiedene Testsituationen
- Ansatz:
 - Elektrischer Weichenmotor schaltet Weiche
 - Stromverbrauch wird gemessen
 - Stromfluss gibt Aufschluss über Schaltbarkeit
 - Polarität gibt Aufschluss über Weichenstellung

Vorgehen:

- Aufbau der Sensoren, Steuerungsmedien, Gleise und Weichen, etc.
- Parallel: Aufsetzen der Entwicklungsumgebung
- Integration notwendiger Tinkerforge-Softwarebausteine
- Manuelle Datenauswertung über grafische Darstellung

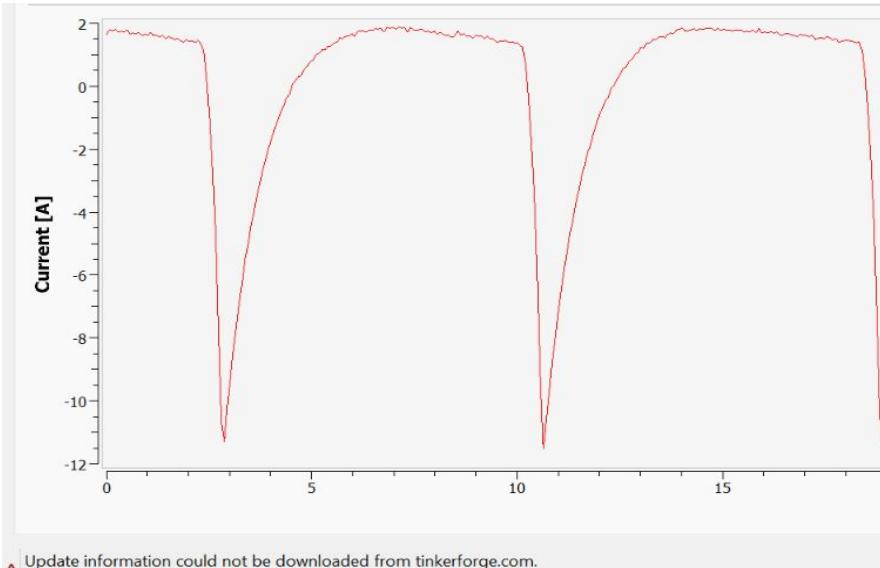
Challenges:

- Zwei verschiedene Messsensoren
- Analoger Sensor ermittelt keine Stromflussrichtung

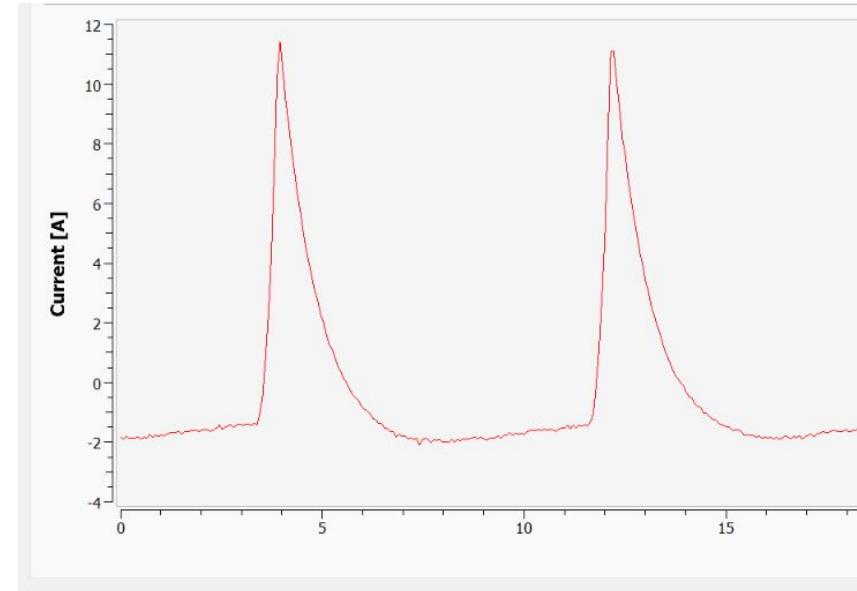
Prozessschritte - Remote Diagnostics

Vorgehen:

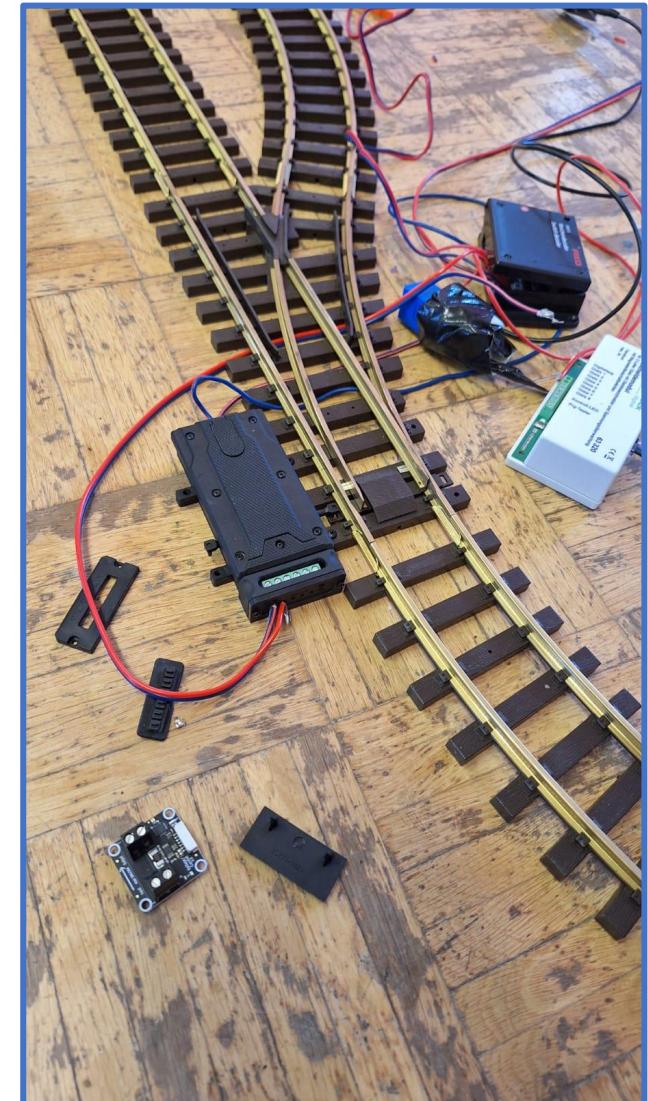
- Analyse der Daten per Python-Skript:
 - Speicherung der Stromfluss-Werte bei Ansteuerung der Weiche in Liste
 - Automatisierte Auswertung per Algorithmus
- Rückmeldung über erfolgreiche Schaltung bzw. Weichenblockade



Stromfluss bei erfolgreichem Umschalten der Weiche



Stromfluss bei blockierter Weiche



Ergebnisdoku - Remote Diagnostics

- Schalten der Weiche (per E-Motor) wird automatisch erkannt
- Messwertanalyse gibt Rückmeldung über Erfolg der Weichenschaltung sowie aktuelle Position bei Weiche 2
- In 20 Testfällen wurde der Zustand 19 mal korrekt erfasst

```
class Measurement:  
    def __init__(self):  
        self.current_event = []  
        self.isWorking = True  
  
    def cb_current(self, _, current, *args):  
        current = abs(current)  
        if current > 50:  
            self.current_event.append(current)  
        else:  
            # abort if there are not enough values in the current event list  
            if len(self.current_event) < 4:  
                return  
            diffs = []  
            # find the biggest change in current  
            for idx, meas in enumerate(self.current_event):  
                if idx+1 < len(self.current_event):  
                    diffs.append(abs(self.current_event[idx+1] - meas))  
  
            # find where in the event it happened  
            index = diffs.index(max(diffs))  
  
            # if it's in the first part of the event, it worked, if not the switch is blocked  
            if (part:=len(self.current_event)/index) < 13:  
                print(f"Weiche hat Endlage erreicht, #Debug: {part}")  
                self.isWorking = True  
            else:  
                print(f"Weiche ist blockiert, #Debug: {part}")  
                self.isWorking = False
```

Aufbau:

- Zwei Langsamfahrstellen: Abzweig (Bahnhof), freie Strecke (Baustelle)
- Integration von NFC-Tags im Gleis
- Sensor unter Flachbettwagen montiert
- Bei Überfahrt kann Tag automatisch erkannt werden und per Wifi an einen Rechner gesendet werden
- Zugsteuerung erfolgt kabellos per LocoNet

Challenges:

- Bei hohen Geschwindigkeiten werden Tags nicht zuverlässig erkannt
 - Lösung: Geringer Timeout für Tag-Erkennung
- Wiederherstellung der Zuggeschwindigkeit nach Ende der Langsamfahrstelle
 - Speicherung von Richtung und Geschwindigkeitsdaten bei Einfahrt ermöglichen Beschleunigung auf Ausgangswerte bei Ausfahrt

Prozessschritte - ATO Digital Signature

```
# Callback function: NFC reader found a new tag
def cb_reader_state_changed(state, idle, nfc):
    global last_tag_id, loconet_socket
    if state == nfc.READER_STATE_REQUEST_TAG_ID_READY:
        ret = nfc.reader_get_tag_id()
        tag_id = " ".join(
            map(str, map("0x{:02X}".format, ret.tag_id)))
    ) # map tag ID to a string

    # check restrictions if the tag was not already read directly before
    if last_tag_id == None or tag_id != last_tag_id:
        print("Found tag of type " + str(ret.tag_type) + " with ID " + tag_id)
        check_construction_restriction(tag_id, loconet_socket)
        check_branch_restriction(tag_id, loconet_socket)

    last_tag_id = tag_id

if idle:
    nfc.reader_request_tag_id()
```

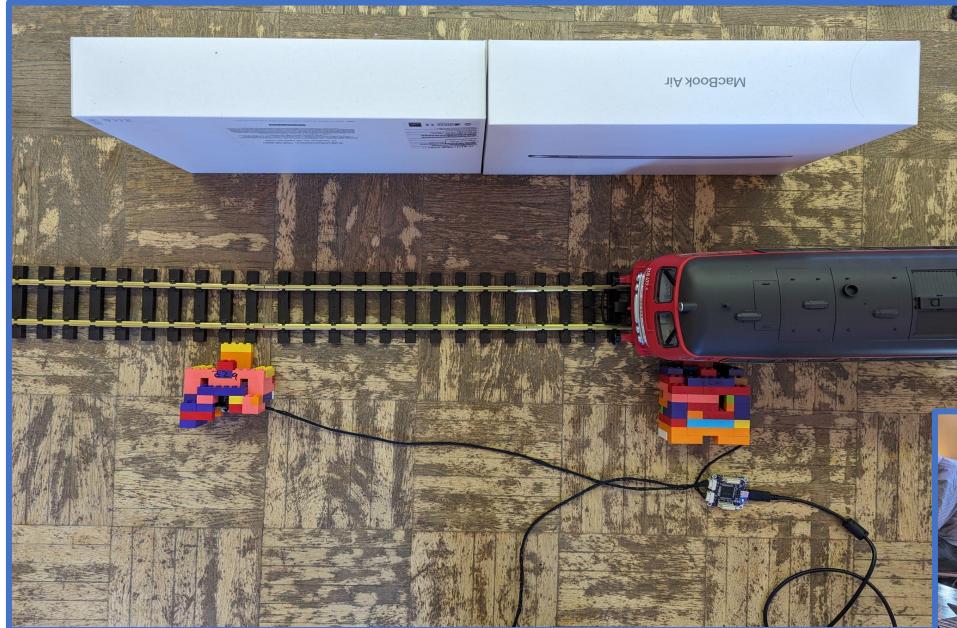
Vorgehen:

- kontinuierliches Scannen nach NFC-Tags
- Speichern des letzten erkannten Tags, um Mehrfacherkennungen zu vermeiden
- wenn neues Tag erkannt: Überprüfen der beiden Langsamfahrstellen (Anfangs- oder Endpunkt erreicht?)
 - Beginn: Reduzierung der Geschwindigkeit
 - Ende: Wiederherstellung der vorherigen Geschwindigkeit

```
# check the speed restriction imposed when entering the branch
def check_branch_restriction(tag_id):
    global loconet_socket, last_speed
    speed, direction = get_speed(loconet_socket)
    if is_branch_start(tag_id, direction):
        print("Entering branch: Start of speed restriction!")
        last_speed = speed
        set_speed(loconet_socket, 20)
    elif is_branch_end(tag_id, direction):
        print("Leaving branch: End of speed restriction!")
        set_speed(loconet_socket, last_speed)
```

```
# check the speed restriction at the construction site
def check_construction_restriction(tag_id):
    global loconet_socket, last_speed
    speed, direction = get_speed(loconet_socket)
    if is_construction_start(tag_id, direction):
        print("Entering construction area: Start of speed restriction!")
        last_speed = speed
        set_speed(loconet_socket, 20)
    elif is_construction_end(tag_id, direction):
        print("Leaving construction area: End of speed restriction!")
        set_speed(loconet_socket, last_speed)
```

- Erkennung von eindeutig identifizierbaren NFC-Tags
 - Automatische Steuerung des Zuges für den jeweils implementierten Abschnitt
- Sehr zuverlässig (6 Fahrten mit jeweils 4 NFC-Tags wurden erfolgreich identifiziert)
- Zugsteuerung funktioniert nur bei zuverlässiger kabelloser Verbindung
- Bei Fehlen eines Tags wird ab dem folgenden Tag die entsprechenden Werte eingestellt



Aufbau:

- Verschiedene Testsituationen und Testdurchläufe
- Ansatz:
 - Abstandssensor misst dauerhaft Abstand
 - Falls gemessene Distanz kleiner als 15 cm wird eine Achse erkannt
 - Kommen weniger Achsen aus dem Bereich heraus als ursprünglich gezählt wurden, wurde ein Wagen verloren

Challenges:

- Filtern von Messfehlern
- Genauigkeit des Sensors (v.a. < 5 cm)
- Messrate des Sensors zu niedrig



Vorgehen:

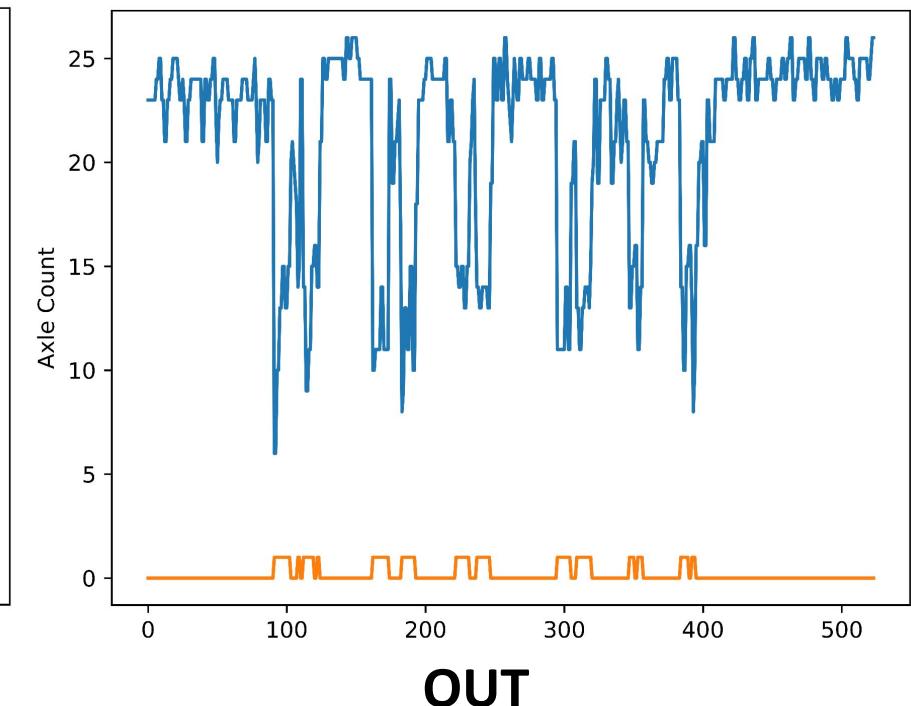
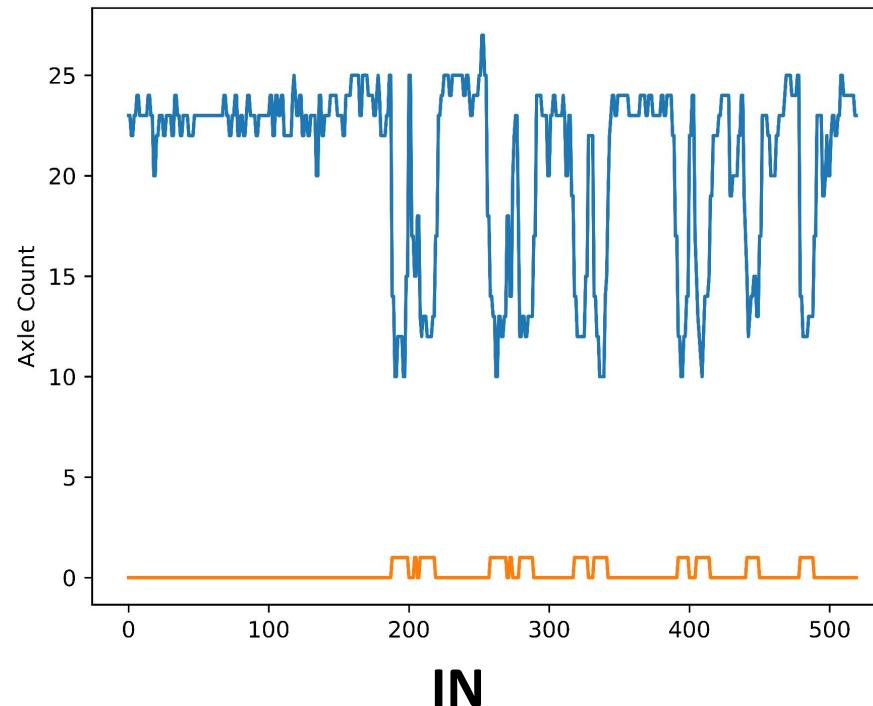
- Aufbau der Sensoren, Steuerungsmedien, Gleise und Weichen, etc.
- Parallel: Aufsetzen der Entwicklungsumgebung
- Integration notwendiger Tinkerforge-Softwarebausteine
- Manuelle Datenauswertung über grafische Darstellung
- Automatische Auswertung der Daten

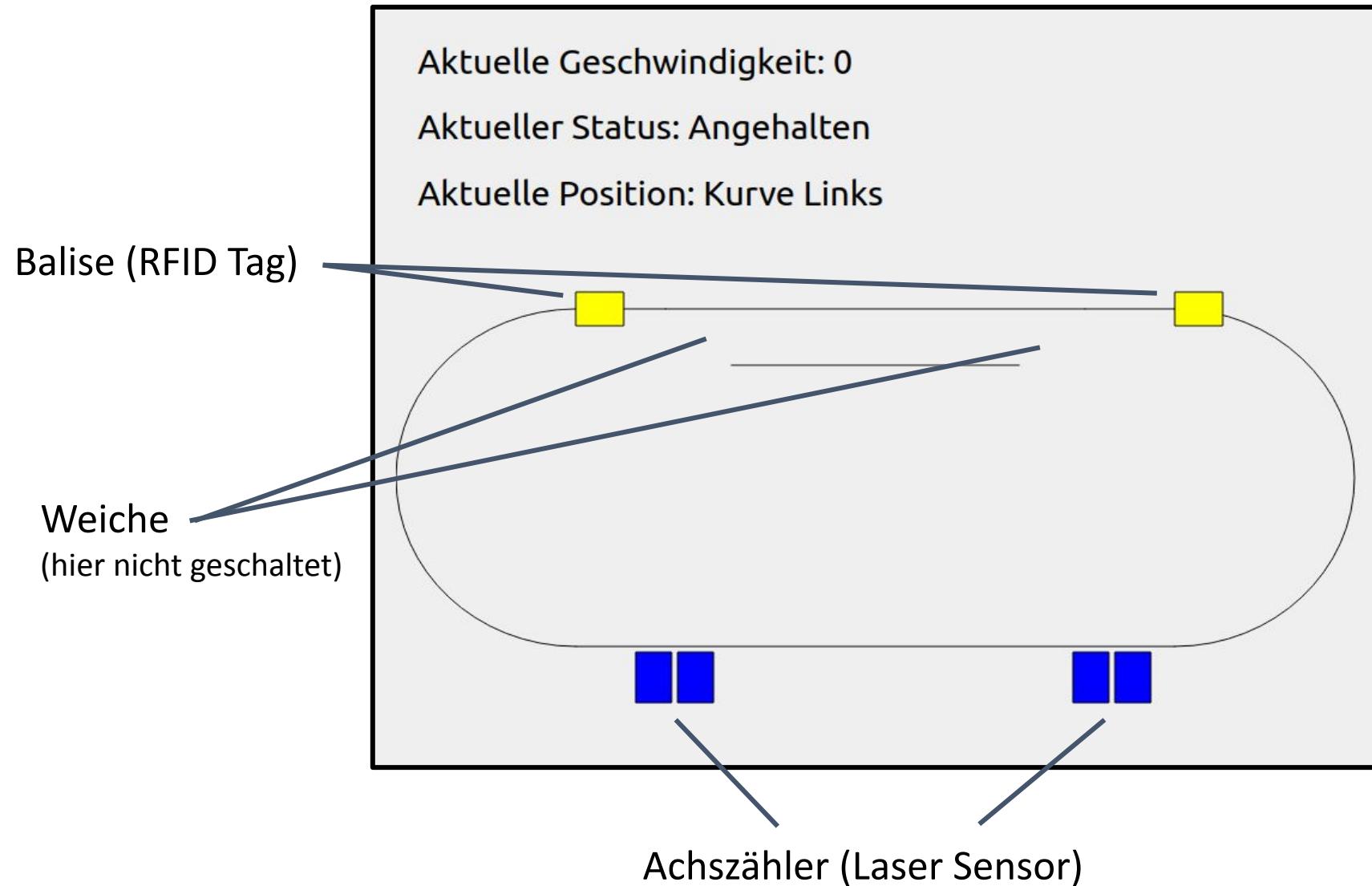
Ergebnisdoku - Steering: Axle Counter

Bekannte Probleme:

- Achszählung für höhere Geschwindigkeiten nicht mehr zuverlässig
 - Messrate der Sensoren nicht schnell genug
- Sehr viel Noise durch ungenaue Sensorik

Beides lösbar durch bessere Sensoren





- Zuverlässige **Erkennung** von **Weichenstörungen**
 - Stopp des Zuges bei blockierter Weiche
- Automatische Erkennung der Weiche (NFC-Tag) und **abbremsen des Zuges** bei **abzweigendem Fahrweg**
- Erkennung der **Langsamfahrstrecke** zur **Achszählung**
 - Zählung der Achsen bei Ein- und Ausfahrt
 - Ausgabe des Ergebnisses über Konsole
- Bestehende **Challenges**:
 - Ausfahr-Achszähler ist unzuverlässig (technische Schwierigkeiten)
 - Weichenposition kann nicht immer identifiziert werden
 - Kein Halt des Zuges bei Verbindungsverlust zum Zentralrechner (konstruktionsbedingt durch Modelllok)
- **Nutzung** des Systems:
 - Zug wird **manuell gestartet**
 - **Weichen** können **per Skript** oder per Fernbedienung geschaltet werden
 - Zug wird **automatisch** weitergefahren und **gesteuert**

- Integration der Projekte
 - NFC-Tags bremsen den Zug zur Achszählung ab
 - NFC-Tags erkennen Weiche zur Abbremsung
 - Achszählung kann mit Weiche kombiniert werden
- Praxisbezug
 - Erkennung der Zugposition mittels Balisen im ETCS-System bereits im Einsatz
 - Achszähler ebenfalls zur Vollständigkeitskontrolle von Zügen an Signalen üblich
 - Zuverlässigkeit zukünftig wichtiger, um mehr Zugverkehr auf dem bestehenden Netz zu ermöglichen

Produzierter Code

Code öffentlich auf GitHub verfügbar

AlexanderBraml / **IoTRailHackathon2023** Public

Code Issues Pull requests Actions Projects Security Insights

main · 1 branch · 0 tags

AlexanderBraml Update Readme 19bb8d5 now 8 commits

File	Description	Time
.idea	Add GUI components	38 minutes ago
gui	Fix GUI demo delay	35 minutes ago
README.md	Update Readme	now
data_logger.py	Update data_logger.py	35 minutes ago
requirements.txt	Add GUI components	38 minutes ago

README.md

IoT Rail Hackathon 2023

Aufgabenstellung

Euer Team bekommt von der Digitalen Schiene Deutschland einen Großauftrag!

Die erste Herausforderung, die euch gestellt wird, ist das Thema der Ferndiagnose einer Weiche.

About

No description, website, or topics provided.

Readme · Activity · 0 stars · 2 watching · 0 forks · Report repository

Releases

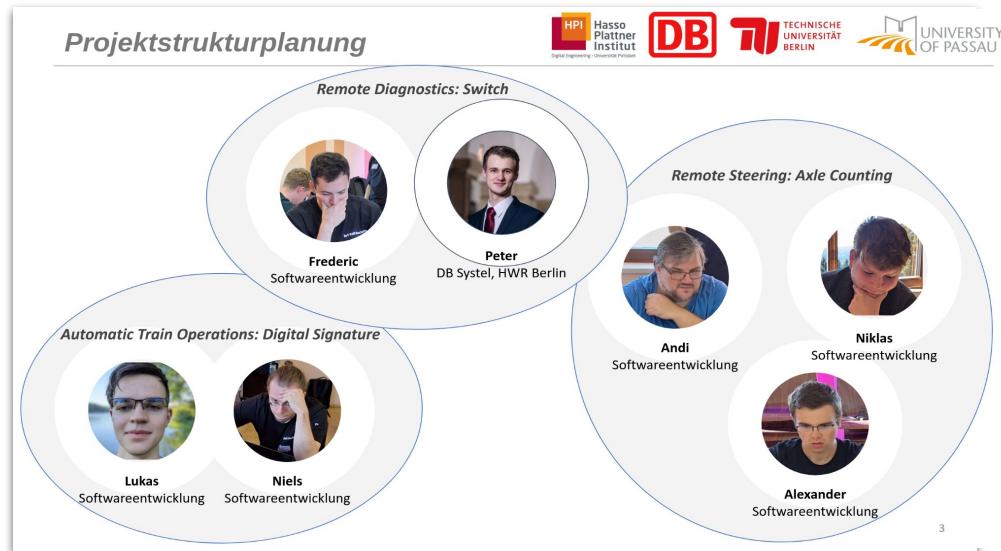
No releases published

Packages

No packages published

Contributors 2

<https://github.com/AlexanderBraml/IoTRailHackathon2023>



Prozessschritte - Remote Diagnostics

Aufbau:

- Verschiedene Testsituationen
- Ansatz:
 - Elektrischer Weichenmotor schaltet Weiche
 - Stromverbrauch wird gemessen
 - Stromfluss gibt Aufschluss über Schaltbarkeit
 - Polarität gibt Aufschluss über Weichenstellung

Challenges:

- Zwei verschiedene Messsensoren
- Analoger Sensor ermittelt keine Stromflussrichtung

Vorgehen:

- Aufbau der Sensoren, Steuerungsmedien, Gleise und Weichen, etc.
- Parallel: Aufsetzen der Entwicklungsumgebung
- Integration notwendiger Tinkerforge-Softwarebausteine
- Manuelle Datenauswertung über grafische Darstellung

5

Prozessschritte - ATO Digital Signature

Aufbau:

- Zwei Langsamfahrstellen: Abzweig (Bahnhof), freie Strecke (Baustelle)
- Integration von NFC-Tags im Gleis
- Sensor unter Flachbettwagen montiert
- Bei Überfahrt kann Tag automatisch erkannt werden und per Wifi an einen Rechner gesendet werden
- Zugsteuerung erfolgt kabellos per LocoNet

Challenges:

- Bei hohen Geschwindigkeiten werden Tags nicht zuverlässig erkannt
 - Lösung: Geringer Timeout für Tag-Erkennung
- Wiederherstellung der Zuggeschwindigkeit nach Ende der Langsamfahrstelle
 - Speicherung von Richtung und Geschwindigkeitsdaten bei Einfahrt ermöglichen Beschleunigung auf Ausgangswerte bei Ausfahrt

8

Prozessschritte - Steering: Axle Counter

Aufbau:

- Verschiedene Testsituationen und Testdurchläufe
- Ansatz:
 - Abstandssensor misst dauerhaft Abstand
 - Falls gemessene Distanz kleiner als 15 cm wird eine Achse erkannt
 - Kommen weniger Achsen aus dem Bereich heraus als ursprünglich gezählt wurden, wurde ein Wagen verloren

Challenges:

- Filtern von Messfehlern
- Genaugkeit des Sensors (v.a. < 5 cm)
- Messrate des Sensors zu niedrig

Vorgehen:

- Aufbau der Sensoren, Steuerungsmedien, Gleise und Weichen, etc.
- Parallel: Aufsetzen der Entwicklungsumgebung
- Integration notwendiger Tinkerforge-Softwarebausteine
- Manuelle Datenauswertung über grafische Darstellung
- Automatische Auswertung der Daten

11