Autonomous Mobile Platforms in Heavy Duty Industry

B. Pulendralingam, C. P. Madsen, D. Palade, E. B. Hansen, P. Debska

Department of Materials and Production, Aalborg University
Fibigerstraede 16, DK-9220 Aalborg East, Denmark
Email: vt2group3120c-f18@m-tech.aau.dk,
Web page: http://www.mechman.m-tech.aau.dk/

Abstract
The fourth industrial revolution, so-called Industry 4.0, brings technologies like Internet of Things (IoT), digitalisation, smart production, autonomous driven vehicles (AGV), etc to the manufacturing market. This paper is written in collaboration with Siemens Gamesa Renewable Energy (SGRE), which is one of the leading players in the production of wind turbines. In order to stay competitive and keep the leading position, SGRE wants to use some of the aspects of Industry 4.0. Therefore, the goal of this collaboration is to investigate how SGRE can improve its production with the use of AGVs. A testing factory located in Brande, Denmark was used for final testing. SGRE has invested in a heavy loaded, semi-autonomous AGV with a payload of 130 tones which requires an operator to be present while running. Due to the complexity of the project and the scale of the heavy loaded AGV, as a hardware MiR100 mobile platform with a payload of 100 kg was used. For further development, data extraction, database, black box, and simulations are investigated. The final product was a MiR100, which was capable of collecting data like battery percentage or localisation, storing them in the database, creating a black box in case of emergency, and drive autonomously. Moreover, a set of simulations validating the possibility of scaling up the solution was delivered.

Keywords: Industry 4.0, AGV, Wind turbine, IoT, Digitalisation, Simulations

1. Introduction

Industry 4.0 has gained increased popularity in the last few years. It is often used as a substitution for a set of new technologies starting the next industrial revolution. According to Boston Consulting Group (BCG) these technologies are autonomous robots, simulation, horizontal and vertical system integration, the industrial Internet of Things (IoT), cyber security, the cloud, additive manufacturing, augmented reality, big data and analytics [1]. The main focus of this paper is to implement some of them in the company to increase its productivity, efficiency and effectiveness, as well as stay competitive and deliver the highest value to their customers. Moreover, the concept of digitalisation is investigated. According to Gartner Inc., which is the world's leading research and advisory company, digitalisation is an array of actions leading to changes in business models and provides new opportunities for producing value [2]. Digitalisation is a key challenge, which most of the industries are facing.

This paper was written in collaboration with Siemens Gamesa Renewable Energy (SGRE) and tested in a pilot facility in Brande, Denmark, using their D3 Generator Mix Line [3]. SGRE, the leader in wind renewable energy, is divided into two main units; onshore and offshore. Further, each of the units is divided into Blades, Nacelle, and Towers divisions. The final product is a combination of three blades, one tower, and one nacelle, see Figure 1. In 2017 SGRE installed 83 GW capacities worldwide [4], supplying energy to around 25 million households. SGRE is keen on new technologies and strives to benefit from available developments to improve their business, and as result stay in the market. Internally in SGRE they developed their own mentality called DATACC (abbreviated from Data Analysed Together Automatic Communication Control). SGRE also puts pressure on optimal workflow, therefore they strive to work according to 5S principle from Lean philosophy. 5S stands for sort, set in order, shine, standardise, and sustain [5].

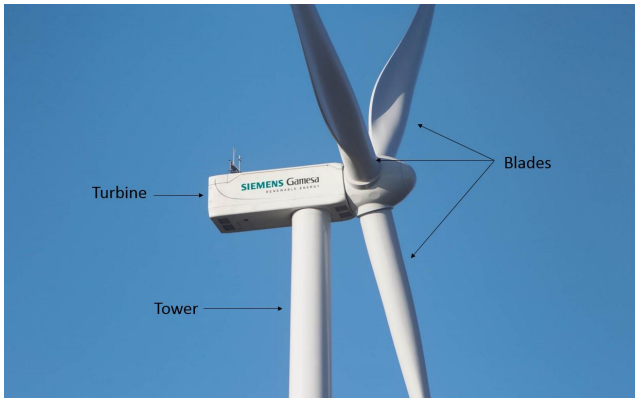

Fig. 1 SGRE onshore windturbine. [6]

1.1 Mobile Platforms

After conducted analysis of the production and interviews with employees, two key bottlenecks not complying with the Lean standards were defined.

One of them was a complicated process of ordering goods from the warehouse. When an operator at the station is missing a part, he has to contact the Team Lead. The Team Lead has to set an order in the system, the order has to be accepted by the main warehouse, palletised and delivered to the production. From the production, it has to be placed in a smaller local warehouse and then transported using a forklift to the requesting station. A whole process can take up to a few hours and requires the involvement of several people. An idea to optimise this procedure was to place the screen connected to the main warehouse at each station, where missing items can be ordered directly and delivered by MiR100 within a few minutes without the involvement of team lead, smaller warehouse, and forklift driver.

The second main issue was a long waiting time of arrival for a heavy load Autonomous Guided Vehicle (AGV). The AGV is used to transport the nacelle between stations and is needed in order to continue the assembly procedure. Knowing the nature and requirements of the problematic tasks AGVs were selected as a potential solution due to the fact that SGRE is already using them.

Among the AGVs, mobile platforms were defined as suitable for solving the problems and will be further considered as a main hardware for the project. For the first bottleneck light load mobile platform with a payload of 100 kilos, MiR100 (see Figure 2), and for the second task high load mobile platform with a payload up to 130 tones, customised wheeled mover produced by Solving were selected.

Fig. 2 MiR100 mobile platform. [7]

1.2 Safety

SGRE has a few leading cultures ensuring the quality of their product and the safety of their employees. The highest priority for them is safety, every new product has to have conducted a risk assessment and hazards definition according to the standard, ISO 12100:2010 (Safety on machinery) [8], which SGRE is certified in. Risk assessment was only conducted for MiR100, as the other mobile platform, Solving, was assessed by Siemens, when they have purchased it. In total five hazards were identified for using MiR100:

1) MiR100 collides with a machine and damages it.
2) During the collision carried parts fall down and can hit an employee.
3) Carried item which was not fastened falls from the MiR100 and can damage another machinery or a human.
4) Human can step onto static MiR100 and fall.
5) A wheel gets stuck or malfunctions causing unexpected movement of MiR100.

Further evaluation of these hazards was conducted by assigning probability level of occurrence according to SGRE's As Low as Reasonably Practicable (ALARP) scale [9]. All of the mentioned hazards were within an acceptable level and therefore are not considered as risks. MiR100 was assessed as safe to be used in the production complying with ISO standards, and internal SGRE standards. Therefore, it is further used as the main hardware for development.

2. Development

For every implementation, there should be a future goal, also known as an ideal case. For this paper, the ideal aspects are focusing on implementing a fully

autonomous mobile platform in a heavy duty industry. There are no official ISO standards to follow, according to mobile platforms in such environment. There are four main aspects to consider. There should be restrictions on the mobile platform to only drive in certain areas, in order to minimise the possibility of collision. The second aspect is the collaborative interaction between mobile platforms and humans to increases safety by minimising the negative awareness in the form of fear. The interaction has two sub-aspects: interfaces and signalling. The last aspect, this paper will take into consideration, is the maintenance through the use of data extraction and gathering. Data has been a growing topic in the manufacturing industry, due to its improvement possibilities through e.g. machine learning. The case presented in this paper gathers data from the mobile platform when a safety stop occurs. The mobile platform could also gather information about its runs, in order to predict future maintenance issues, also minimising the future breakdowns, potentially endangering the employees.

2.1 Data Extraction

Because MiR100 has the Robotic Operation System (ROS), which is a collection of tools, libraries and conventions specifically aimed to simplify the information sharing [10], the implementation of a data extractor was done by creating a python program. It runs on the MiR100's roscore as a node, using a rosbridge [11] by specifying the IP as: mir_api_url=http://192.168.12.20:8080/v1.0.0/. The collected data is presented in Table I.

Environment Map	**Path Planning**
Map (**String**) Position X (**Float**) Position Y (**Float**) Position θ (**Float**)	Linear Velocity (**Float**) Angular Velocity (**Float**) Linear Acc. X (**Float**) Linear Acc. Y (**Float**) Linear Acc. Z (**Float**) Distance to target (**Float**)
Communication	**Main Hardware**
Robot Connected (**Bool**) BT Connected (**Bool**)	Battery % (**Float**) Battery left (**Int**) Battery prediction (**String**) Uptime (**Int**) Errors (**String**)
Misc	
Date and time (**String**) Robot name (**String**) Status (**String**) Status ID (**Int**)	

Tab. I All the elements of the collected data.

Due to fast sharing capabilities using REST API [11], the data is collected each second. A simple example of extracting the information of the *status* from MiR100, is presented in Listing 1.

```
{def get_current_status(update,
    arg=None):
  global status_response
  if not update:
    return status_response[arg]
  else:
    try:
      status_response =
          requests.get(mir_api_url+"status",
          timeout=timeout).json()
      return True
    except Exception as e:
      rospy.logerr("Error
          get_current_status(): %s", e)
      return False
while not rospy.is_shutdown():
    start = time.time()
    ts = time.localtime()
    if (get_current_status(update=True)):
      data_dict["connected"] = True
    else:
      data_dict["connected"] = False
      rospy.logwarn("Unable to get
          MiR100 status information")
    #* Misc
    data_dict["robot_name"] =
        get_current_status(update=False,
        arg="robot_name")
    data_dict["status"] =
        get_current_status(update=False,
        arg="state_text")
    data_dict["status_id"] =
        get_current_status(update=False,
        arg="state_id")
    data_dict["linear_vel"] =
        get_current_status(update=False,
        arg="velocity")["linear"]
    data_dict["angular_vel"] =
        get_current_status(update=False,
        arg="velocity")["angular"]
```

Listing 1 Python commands for accessing data on MiR100

2.2 Black Box

Black box represents a collection of vast data collected in case of an emergency which is used to recreate the details of the accident, similar to an aircraft black box [12]. In ROS environment there is a file called *bag* which stores ROS messages such as map, path and laser-scanners, that afterwards can be played on an external

program such as *rviz*. The next topics from ROS were selected specifically for compatibility with rviz:

/tf
: Transformation frames

map
: The current map

b_raw_scan
: The laser-scan data from the rear laser-scanner

f_raw_scan
: The laser-scan data from the front laser-scanner

move_base_node/SBPLLatticePlanner/plan
: Topic containing the global path from start to goal

move_base_node/MIRPlannerROS/visualization_mark
: The local path planning of the MiR100

move_base_node/local_costmap/robot_footprint
: A blow-up footprint of the MiR100. The robot will not move closer to anything infiltrating this footprint

odom_comb
: Provide the position of MiR100

camera_floor/depth/image_rect_raw
: The depth camera mounted in the front of the MiR100

The program called *bagger* was developed in Python and implemented as a ROS node in the roscore of MiR100. Because the amount of data gathered is vast it is impossible to gather everything at the exact moment an emergency occurs. Some data is constantly collected and then rewritten every five minutes, like laser-scan, and other data is collected only once at the robot turn on, such as the transformation frames. The gathering of data normally occurs with the frequency of 25 Hz to enable a seamless playback, but because of a quickly growing size of the file (60 seconds of bagging can reach 500 MB) it was decided that some topics will be collected more rarely, for example, the map will be collected each 20 seconds.
To keep a small file size compression is required. LZ4 compression type was deemed best for the current problem since it permits a relative easy playback from the computational point of view and creates an acceptable file size, around 240MB for 60 seconds of recording, see Figure 3.

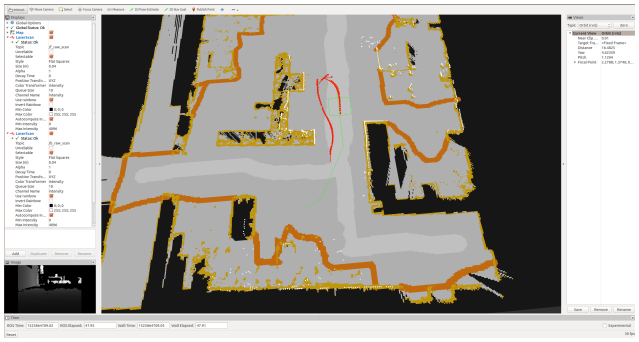

Fig. 3 Playback from a bagger file using *rviz*.

In order for the bagger functionality to work properly, the program needs to know when MiR100 enters an emergency mode. This is done by subscribing the dataExtractor node to a topic called *light_cmd*. When dataExtractor receives the message about emergency it sends a service call to *mir_custom_bagger* as in Listing 2 specifying the event and time. The *bagger* then saves the file giving it a name in format **EVENT_DATE_TIME**.bag.

```
{def save_bag(req):
    global bag
    rospy.loginfo("Service
       mir_bagger_save_bag called,
       locking bag")
    with lock:
        rospy.loginfo("Bag locked!")
    try:
        rospy.loginfo("Bag released")
        script_path =
           os.path.abspath(__file__)
        script_dir =
           os.path.split(script_path)[0]
        real_path = '__temp_bag__.bag'
        ts = time.localtime()
        date =
           time.strftime("%d.%m.%y",ts)
        time_stamp = time.strftime("%X",
           ts)
        prior_file_path =
           os.path.join(script_dir,
           'BAG/PRIOR.bag')

        if
           os.path.exists(prior_file_path):
          # create new path and a prior.
           return
              SavedBagResponse(True,
              f_new_part2)
        else:
           # create only new path
           return
              SavedBagResponse(True,
```

```
                f_new)
    except Exception as e:
        rospy.logerr("OS filesystem
            operations error: %s", e)
    return SavedBagResponse(False, None)]
```

Listing 2 Python commands for calling bagger service.

To run a playback of the black box follow the procedure:

Terminal 1:
Start a ROS Master with *roscore*

Terminal 2:
Launch rviz with *rviz*

Terminal 3:
Navigate to the location of the bag file

Terminal 3:
Play the bag file with *rosbag play* and the name of the bag file

2.3 Database

The extracted data has to be stored and organised in a logical way using databases. For a purpose of this paper, a free commercialised database compatible with ROS and Linux was defined as a requirement. Requirements were ensuring that database can be integrated with the dataExtractor node. Database complying with the requirements and chosen for final implementation was SQLite3. The database development was a direct implementation of transferring the data from the received JSON file (see Listing 3) to SQLite3 database.

```
{"battery_percentage": 76.3,
    "bt_connected": false, $"linear_vel":
    0.0, "robot_name": 'mir_r01',
    "angular_vel": 0.0, "uptime": 9345,
    "errors": [],
    "battery_empty_at_time": '21:06:48',
    "distance_to_next_target": 0.0$,
    "time_stamp": '12:56:15', "status":
    'ManualControl', "map": '465 map',
    "status_id": 11, "linear_acc_x":
    0.0010000000474974513$, "pos_theta":
    -176.56, "linear_acc_z":
    1.0010000467300415, "linear_acc_y":
    0.004000000189989805,
    "battery_left_sec": 36633,
    "connected": true, "date":
    '04/13/18', "pos_x": 20.24, "pos_y":
    -3.32,
    "data_extraxt_durentation_sec":
    0.08661890029907227}
```

Listing 3 The saved JSON message.

The biggest challenge was to map the correct entries type to the types available in SQLite3. According to database standards, entries in the database have assigned the following types: *robot_name* text, *status_id* integer, *date* numeric, *linear_acc_x* real, *time_stamp* numeric, *pos_theta* real, *battery_percentage* real, *linear_acc_z* real, *linear_vel* real, *battery_left_sec* integer, *angular_vel* real, *battery_empty_at_time* numeric, *uptime* integer, *connected* text, *errors* text, *pos_x* real, *distance_to_next_target* real, *pos_y* real, *status* text, *data_extract_duration_sec* real, *maps* text. Each entry has a specific data type which ensures that the conversion to the database is correct. Final database is shown in Figure 4.

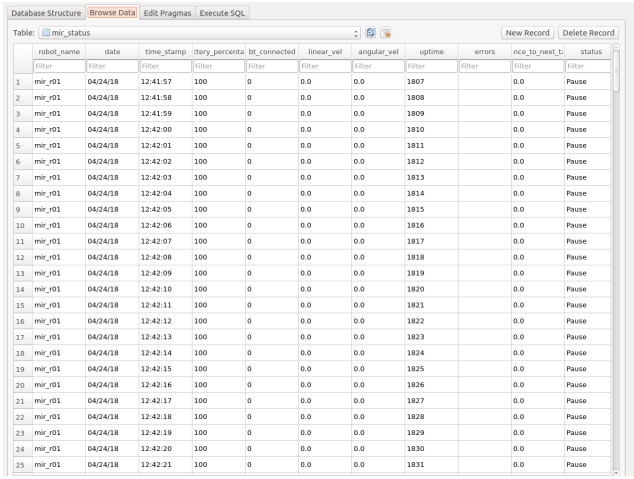

Fig. 4 An example of database in SQLite3.

According to the company's suggestion, the first displayed information is the name of the machinery, since it is the most important. For MiR100 it is *mir_r01*.

In order to run the database a network connection to MiR100 is established, a node for data extractor to receive the data and a package for listening are run.

2.4 Integration of Technologies

To ensure seamless integration of developed technologies, like data extraction, black box, and database a single-board computer, Raspberry Pi, was added. All of the ROS nodes for each technology were uploaded. Raspberry Pi also serves as a hardware for the added button to MiR100 as a release function, to give a signal to the platform that it can continue the task after being paused.
From a network point of view each of the devices: MiR100 router, Raspberry Pi, and MiR100 have static IP addresses. Moreover, a connection is wired to ensure

a stable signal and low latency. Network connection together with assigned IP addresses is shown in Figure 5.

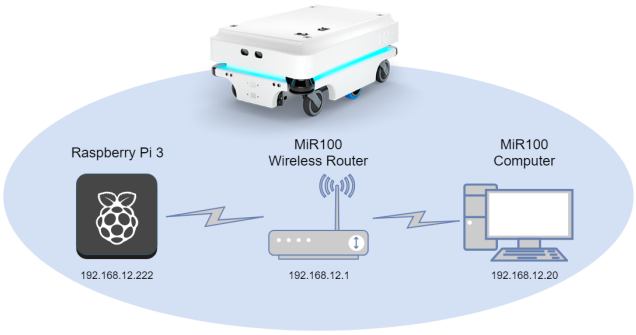

Fig. 5 The network layout.

In the final product, four nodes; dataExtractor, button driver, black box and database are connected to MiR100 ROS Master, see Figure 6.

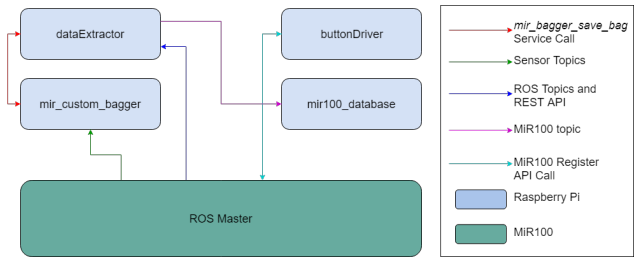

Fig. 6 The connections between ROS nodes and ROS Master.

Previously mentioned nodes were run using a single launch file, see Listing 4.

```
<launch>
<node pkg:"dataExtractorMiR100"
    type="mir100_data_extractor.py"
    name="mir100_data_extractor"/>
<node pkg:"dataExtractorMiR100"
    type="data_base.py" name="data_base"/>
<node pkg:"mirBagger"
    type="mir_bagger.py"
    name="mir_bagger"/>
<node pkg:"dataExtractorMiR100"
    type="button_driver.py"
    name="button_driver"/>
</launch>
```

Listing 4 Launch file for starting all of the nodes.

Integrated product was further used for testing.

2.5 Simulations

Simulation is a powerful tool to predict if a new solution is beneficial and to aid in product and/or production line development. The discrete-event simulation is done to investigate the need for mobile platforms. Results show that transfer times represent 18.18% of the total time needed for assembling the nacelle for a wind turbine.

The second type of simulation is a physical simulation creating the foundation for a digital model of the MiR100. This model could in the future be used to evaluate the implementation of a MiR100 in various settings before it is tested and implemented, thereby reducing potential cost and interference in production. This physical simulation is done in the simulation tool V-Rep, which has many capabilities including the ability to use the existing software on the MiR100 [13]. However, for the purpose of this paper, only a very basic proportional robot controller following a path in a simulation environment of the production facility is presented. In order to make the simulation as smooth as possible, only the wheels of the MiR100 are dynamic. Only the two actuated wheels move the robot while the rest are static. These two wheels have an implemented PID controller that takes a rotational velocity as input. The path planning used for creating paths from the start configuration to the goal configuration is made using the Open Motion Planning Library (OMPL) in V-Rep, which is a collection of sampling-based path planning algorithms [14]. In general, sampling-based algorithms work by checking random non-colliding points in the map and connecting them. Given enough iterations, they will always find a path if one exists, and possibly also the optimal path [15]. The task for the robot controller is to chase a point moving along the path [16]. The MiR100 is a differential-drive robot, in which all actuated wheels are independent. The forward velocity, V is given by:

$$V_{robot} = \frac{V_{left} + V_{right}}{2} \tag{1}$$

and the rotational speed is given by

$$\omega_{robot} = \frac{V_{left} - V_{right}}{d} \tag{2}$$

where d is the distance between center of the wheels on the same axis. Given a desired velocity and rotational speed the velocities of each wheel is:

$$V_{right} = V_{robot} + \frac{d}{2} \cdot \omega_{robot} \tag{3}$$

$$V_{left} = V_{robot} - \frac{d}{2} \cdot \omega_{robot} \tag{4}$$

The rotational velocity of each wheel is then given by

$$\omega_{wheel} = \frac{V_{wheel}}{r} \tag{5}$$

where r is the wheels' radius.
As this is a proportional controller, the desired velocity is 0.5 times the distance to the point on the path and the desired rotational velocity is 0.1 times the angle to the point on the path. These velocities are used to find the velocity of each wheel using Equations 3 and 4, which are then used to find the rotational velocities for each wheel using Equation 5. In Figure 7 the black line is the planned path and the magenta line is the path of the robot when moving from the upper right corner to the position on the left. As can be seen, the planned path does not take into consideration the turning radius and the robot understeers. Furthermore, the robot controller has no stopping criterion, does not take into consideration the pose of the goal configuration and cannot move backwards.

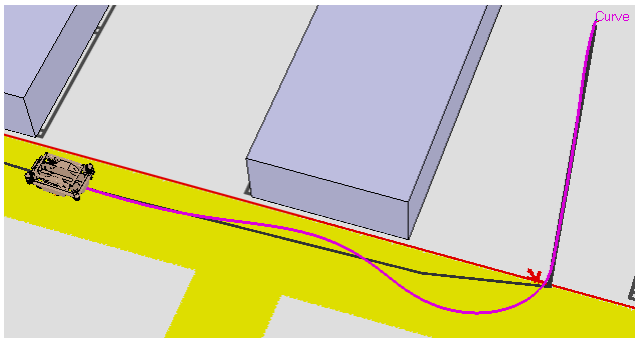

Fig. 7 The path of the robot (magenta) next to the planned path (black).

3. Results

To verify the developed solution several tests were conducted. All of them had the same premise but were conducted under different circumstances. The first test was conducted in a lab at Aalborg University, the second one in the SGRE's D3 Generator Mix Line in Brande during production-off day, where no employees or moving machinery were present. Third and the last test was the closest to the real environment, where employees were working as usual. The two first tests were successful in verifying the functionality of the developed hardware and software.

Fig. 8 The route at SGRE smart production. The route is 101 metres.

In the third test, MiR100 drove the route visualised in Figure 8 ten times in each direction. Half of the runs were with normal production and the last half was with at least one planned blockage of the route. This was done to see how long extra the transportation time would be. It could be seen that the runs with no interrupts had an average time of 160 seconds and the ones with planned interrupts had an average time of 166 seconds. From this, it can be concluded that each interrupt adds around 6 seconds to the transportation time. Furthermore, in the fourth run from Robot cell to Origin, the MiR100 was manually put into emergency stop out of concern for a collision. This action activated the bagging service of the black box, which was examined to see what MiR100 saw and what it had planned to do. In Figure 9 an image from the black box is shown. From the black box, it can be seen that MiR100 can see the forklift as well as the forks themselves. Furthermore, there is no indication of MiR100 trying to go closer or through the forklift and therefore it can be concluded that the manually emergency stop was an unnecessary decision. Moreover, this incident also showcases the actual use of the black box, namely the investigation of an incident.

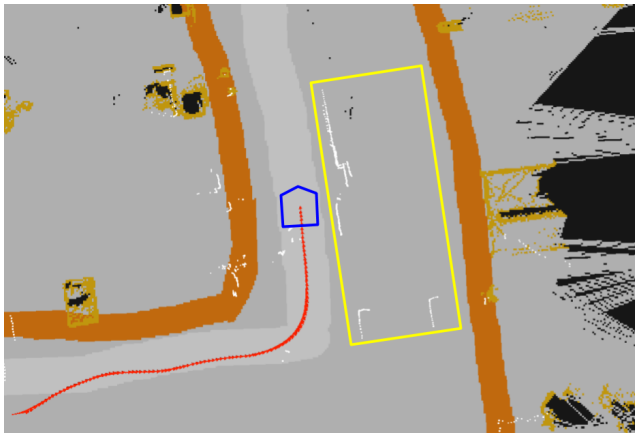

Fig. 9 MiR100's black box at the time of the emergency stop. The yellow square highlights the forklift, the blue polygon is the MiR100 and the red lines are the odometry data. On the map, the grey is the floor, light grey is preferred path, dark orange is prohibited area and yellow is walls. The white dots are the laser-scanner values.

4. Discussion

This paper focused on building the foundation for implementing an autonomous mobile robot in SGRE facility in Brande. With the Industry 4.0 aspect and the guidelines from SGRE, the following development focuses were made: *Data extraction*, *Black box*, *Database*, *Release button*, *Integration* and *Simulation*.

In order to be critical of what data is important, the data extraction aspect was based on the company's needs, but also taking into consideration which data would be required for development. The data extraction was structured into six different categorise acting as the baseline. This means that if another company with different requirements or orientations would want to implement a mobile platform, they could use those categories as the baseline. Out of the six categories, only *Motor Controller* was not collected. Even though important data such as velocities was acquired, potential torque and detection of wheel slip could be useful data according to the maintenance of MiR100.

Safety is an issue in this given case and in general an important aspect of the implementation of new technology. To complement the already implemented safety capabilities of MiR100, a black box was added. During the development of the black box certain problems arose regarding the collection of the data. Especially the laser-scan data along with its transformation frames were out of sync.

The database was developed using SQLite3. SQLite3 misses the date and time type, meaning some functionality with an easy reading of entries would disappear this could not be a problem if an enterprise solution like SQL was used. Moreover, the database stores all of the data in one table called *status*. This will lead to a large database table, e.g. for a working week of 40 hours with a collection of 1 entry per seconds, will lead to: $60 \cdot 60 \cdot 40 = 144\ 000$ entries. Therefore, it could be better to create a new table for each day or week in order to structure the database more thoroughly.

For the user interaction with MiR100, a push-button was utilised to execute next task. In order to improve the functionality the implementation of a GUI with instructions and visualisations could be helpful.

Integration of the ROS nodes consisted of three parts: *hardware integration*, *network layout* and *runtime usage*. The Raspberry Pi's integration with MiR100's onboard computer and the network brought some extra complications in the form of network scope and computing power (constant CPU load above 80%). The Raspberry Pi gave warnings of low voltage, though no direct impact was noticed. Nonetheless, a more powerful machine with more disk capacity would have been beneficial, because of the quickly growing black boxes and database files.

Simulations were used to investigate the Production Line with discrete-event simulation. Also, a physical simulation was made, to simulate MiR100 in the production. The discrete-event simulation was build upon the VSM and information from SGRE, which lead to the issue that no distinction between value adding and non-value adding tasks. This gives a misleading VSM while beneficial information is lacking. The physical simulation was made in V-Rep where a CAD model of MiR100 was imported. The MiR100 was combined with a built-in path planning algorithm in order to make a simple simulation of MiR100's movement. Ideally, MiR100's own software would be implemented in the simulation for a more accurate representation.

5. Conclusion

Even though the development was done on a light load mobile platform, MiR100, it showed a great potential within the field of data gathering and how it draws the company closer to the concept of digitalisation and AGVs. Moreover, a fully functional database, release button, and black box were integrated and run successfully on a single-board computer, Raspberry Pi 3. A simple simulation model of the MiR100 was

created and could upon further development be useful for validating the actual implementation of mobile platforms. A test in SGRE D3 Generator Mix Line was carried out to investigate the behaviour of the mobile platform in a fully operating heavy duty production line. The test showed that MiR100 is capable to navigate and handle such an environment.

Acknowledgement
The authors of this work gratefully acknowledge Grundfos for sponsoring the 6th MechMan symposium.

References

[1] M. Rüßmann, M. Lorenz, P. Gerbert, M. Waldner, J. Justus, P. Engel, and M. Harnisch, "Industry 4.0: The future of productivity and growth in manufacturing industries," Boston Consulting Group, vol. 9, Apr. 2015.

[2] Gartner Inc., "Gartner it glossary: "digitalization"." Website, 2018.

[3] Siemens Gamesa Renewable Energy, "United, in shaping the renewable energy industry." Website, 2017.

[4] Siemens Gamesa Renewable Energy, "Expertise and global track record." Website, 2018.

[5] T. Ohno, "Taichi ohno's workplace management.," Special 100th birthday edition. New York: McGraw Hill, p. 112112, 2013.

[6] Siemens-Web, "Press photes." Website, 2017.

[7] Mobile Industrial Robots ApS, "MiR100^{TM}," 2016-2017.

[8] International Organization for Standardization, "ISO 12100:2010 Safety of machinery – General principles for design – Risk assessment and risk reduction," ISO, 2013.

[9] B. Henriksen, "Information from siemens regarding safety, digitalization and behaviour in production." February 2018.

[10] ROS site, "About ROS," May 2018.

[11] MiR-Support, MiR robot REST api. Mobile Industrial Robots, version 1.0.0 ed., Apr. 2018.

[12] R. Thomas, "The black box flight recorder," May 2018.

[13] Coppelia Robotics, "V-rep physical robot simulation," 2018.

[14] I. A. Şucan, M. Moll, and L. E. Kavraki, "The Open Motion Planning Library," IEEE Robotics & Automation Magazine, vol. 19, pp. 72–82, December 2012. http://ompl.kavrakilab.org.

[15] H. Choset, S. Hutchinson, K. Lynch, G. Kantor, W. Burgard, L. Kavraki, S. Thrun, and R. Arkin, Principles of Robot Motion: Theory, Algorithms, and Implementation. A Bradford book, MIT Press, 2005.

[16] N. Kummer, "Path planning with vrep," July 2014.