Integration of an MES and AIV Using a LabVIEW Middleware Scheduler Suitable for Use in Industry 4.0 Applications

Muzaffar Rao *, Liam Lynch, James Coady, Daniel Toal † and Thomas Newe ‡

Centre for Robotics and Intelligent Systems (CRIS), Department of Electronic and Computer Engineering, University of Limerick, Limerick V94 T9PX, Ireland; liam.lynch@ul.ie (L.L.); james.coady@ul.ie (J.C.); daniel.toal@ul.ie (D.T.); thomas.newe@ul.ie (T.N.)

* Correspondence: muzaffar.rao@ul.ie

Received: 20 August 2020; Accepted: 9 October 2020; Published: 11 October 2020

Featured Application: This work provides details of an integration of a mobile Autonomous Intelligent Vehicle (AIV) with a Manufacturing Execution System (MES), which helps to minimize human intervention requirements in a manufacturing process. Reduction in human intervention in the manufacturing process is one of the key requirements of Industry 4.0. The proposed integration technique helps to ensure efficient production and provides optimum use of the production cells in a smart manufacturing environment.

Abstract: Industry 4.0 uses the analysis of real-time data, artificial intelligence, automation, and the interconnection of components of the production lines to improve manufacturing efficiency and quality. Manufacturing Execution Systems (MESs) and Autonomous Intelligent Vehicles (AIVs) are key elements of Industry 4.0 implementations. An MES connects, monitors, and controls data flows on the factory floor, while automation is achieved by using AIVs. The Robot Operating System (ROS) built AIVs are targeted here. To facilitate MES and AIV interactions, there is a need to integrate the MES and the AIVs to help in building an automated and interconnected manufacturing environment. This integration needs middleware, which understands both MES and AIVs. To address this issue, a LabVIEW-based scheduler is proposed here as the middleware. LabVIEW communicates with the MES through webservice interactions and has support for ROS. The main task of the scheduler is to control the AIV based on MES requests. The scheduler developed was tested in a real factory environment using the SAP MES and a Robotnik ‘RB-1’ robot. The scheduler interface provides real-time information about the current status of the MES, AIV, and the current stage of scheduler processing. The proposed scheduler provides an efficient automated product delivery system that transports the product from process cell to process cell using the AIV, based on the production sequences defined by the MES. In addition, using the proposed scheduler, integration of an MES is possible with any low-cost ROS-built AIV.

Keywords: industry 4.0; smart manufacturing; AIVs; MES; LabVIEW; Turtlebot; RB-1; scheduler

1. Introduction

The term ‘Industry 4.0’ initially originated from Germany, and it represents the fourth industrial revolution [1]. Industry 4.0 is still visionary but realistic [2] as technology is changing faster than ever. A summary of the industrial revolutions [3–6] is given below in Figure 1.
Autonomous Intelligent Vehicles (AIVs) are critical components of Industry 4.0 [8,11] in order to plant floor systems and provides real-time information exchange for faster and better decision making. This is because the MES acts as a bridge between different operations. By connecting AIVs to a programmable logic controller, central server, or database, the actions of robots can be monitored and controlled. This leads to smarter decisions, better ability to predict future needs, better-designed products, and more efficient use of resources. Connected technologies transform the way products are developed, and this enables rapid prototyping and testing by adding connectivity to previously unconnected products and leads to new products and services. The goal of Industry 4.0 is the “smart factory” with cyber-physical systems capable of autonomously exchanging information, triggering actions, and controlling each other independently. Technologies that form building blocks of the Industry 4.0 [8–10] are Big data and analytics, Industrial Internet of Things, Cybersecurity, Cloud computing, Additive Manufacturing, Simulation, Augmented Reality, and Cyber-Physical Systems. In addition, Manufacturing Execution Systems (MESs) and Autonomous Intelligent Vehicles (AIVs) are critical components of Industry 4.0 [8,11] in order to achieve interconnection and automation objectives. This work focuses on the integration of these two components.

An MES [12], is an information system that connects, monitors, and controls data flows on the factory floor in real-time. This system will act as an enabler to attain the end to end digitization and interconnectivity objectives of Industry 4.0. This is because the MES acts as a bridge between different plant floor systems and provides real-time information exchange for faster and better decision making.
All resources involved in the production process can be interconnected using the MES to support a smarter manufacturing process.

AIVs are used across countless industries, including manufacturing [13]. Companies that aim to have a smart factory to facilitate Industry 4.0 must integrate robotics into its operations. By connecting AIVs to a programmable logic controller, central server, or database, the actions of robots can be automated and coordinated. AIVs can complete tasks intelligently and in an organized manner with minimal human input. Materials can be transported across the factory floor using obstacle avoiding AIVs, coordinating with fleet mates, and pinpointing where pickups and drop-offs are needed.

The above discussion helps the reader to understand that the Industry 4.0 vision of ‘smart factory’ is not possible without the deployment of a MES and AIVs. As mentioned earlier, this work focuses on the integration of an MES and AIV (as shown in Figure 2), which is necessary to build a fully connected, automated smart manufacturing environment. The Robot Operating System (ROS) is an open-source platform which consists of a set of software libraries and tools that help to build robot applications. To integrate an MES with an AIV, a middleware is needed to perform the command translations in order for the MES and ROS-based AIV to communicate. Here, a LabVIEW-based scheduler is developed as the middleware. LabVIEW communicates with the MES through webservices and also has support for ROS. The scheduler development is divided into three main parts for: (a) LabVIEW-MES communication, (b) LabVIEW-AIV communication, and (c) the Main scheduler. In part (a), the scheduler imports MES webservices and generate equivalent LabVIEW VIs. The generated LabVIEW VIs are edited and configured using the .NET functions of LabVIEW. In part (b), the scheduler uses the LabVIEW add-on ‘ROS for LabVIEW’ to establish communication with an AIV. Part (c) consists of one complete cycle of scheduler operation, which involves eight steps. These eight steps include checking of pickup locations status, selection of one pickup location, pick the product, confirm to the MES about completion of pickup operation, checking of drop-off locations status, selection of one drop-off location, drop the product, and confirm to the MES about the completion of the drop-off operation. The scheduler was tested in two phases, initially, the Turtlebot-3 (burger model) [14] robot was used for testing during scheduler development and finally, the scheduler was commissioned using the Robotnik ‘RB-1’ [15] robot in a real factory environment. The scheduler supports three pickup and three drop-off locations. The pickup and drop-off locations are selected based on predefined priority rules. The developed scheduler interface is user friendly and can be easily used by a non-technical operator and it supports data logging (of each step of scheduler operation), as well as displays a live log of operation.

![Figure 2. Proposed Manufacturing Execution System (MES) and Autonomous Intelligent Vehicle (AIV) integration.](image)

Figure 3 below shows the overall architecture of the designed system. As mentioned previously, the scheduler is designed to sit as middleware between the MES and the AIV. Its function is to perform...
command/feedback conversions so that the MES can communicate with the AIV. Figure 3 describes at a high level, the system, and its method of operation. More in-depth and detailed information on the method of operation can be found in Section 3. The system is developed around the Scheduler, which acts as the conversion tool for all communication between the MES and AIV. At the top level, resides the MES, the brain of the factory. The communication between SAP MES and production equipment is provided by the Equipment Connector (ECo). This simplifies the interface of production equipment to an MES. The ECo is used in the standard SAP netweaver environment and uses the netweaver stack features like deployment, monitoring, logging, and internal database. The ECo communicates through webservice with the MES. For the integration of production equipment which does not have a webservice interface, an Equipment Protocol Adapter (EPA) is used, which translates the equipment specific protocol (e.g., XML, SPS, file transfer) to the ECo webservice.

![Figure 3. System architecture.](image)

Here, the scheduler-MES and scheduler-AIV communication are in real-time; therefore, if something goes wrong during scheduler operation, then the scheduler will detect it at run-time and will indicate the error using flag indicators on the front-panel of the scheduler. In addition, in terms of the time delay, the scheduler-MES communication is in millisecond while scheduler-AIV communication takes seconds (to reach the destination and in terms of auto-docking). Timing details of each step of scheduler execution is given in Section 4 in Table 1.

| Step | Time (ms) |
|------|-----------|
| Step-1 | 1570 |
| Step-2 | 29,222 |
| Step-3 | 69,407 |
| Step-4 | 1693 |
| Step-5 | 1332 |
| Step-6 | 30,308 |
| Step-7 | 83,271 |
| Step-8 | 1888 |

A lot of work has been published on robotic automation [16–20] and MES [21–25]. Relevant discussion to this work is published in Reference [26–29], but, to the best of the authors’ knowledge, the detailed practical solution for the integration of an MES and an AIVs is not previously published. Some manufacturers of commercially available AIVs [30,31] offer high-end developed solutions, including MES integration, but with license restrictions. The integration solution provided here can help the manufacturing industry to develop its own high-end system using low-cost AIVs. An overview...
of the previously published relevant work in Reference [26–29] is included here. In Reference [26] authors have focused on the challenges associated with AIV integration with MES in a manufacturing environment. This work mentioned the need for an interface tool, which can communicate with AIV and MES webservices; however, there is no discussion on how to develop the interface tool. A review of low-cost AIVs for advanced manufacturing systems is provided in Reference [27]. This work recommends the off the shelf low-cost AIV to integrate it into intralogistics. The use of ROS is also recommended here to build a low-cost solution. This work highlights that low-cost AIVs like Robotnik has no communication support with an MES. In Reference [28] the authors focused on data mining methods to integrate them into an MES. The data analysis was made with a focus on the Baxter cobot. In Reference [29] the authors presented a specific EU project STAMINA, which involved the integration of a cognitive robot into the manufacturing process. Here, a logistic planner was used to link an Enterprise Information System (EIS) with a robot. The authors defined the logistic planner as a mechanism responsible for integrating the STAMINA system with the IES. The technical details of the logistic planner was not provided.

The LabVIEW-based scheduler proposed in this work is unique. This work focuses on the scheduler development details, while other parts of the project like MES webservices development and lift system (mechanical part) to pick and drop the product were completed by other project partners and are out of scope for discussion here.

The paper is organized as follows. Section 2 briefly described MES and AIVs, Section 3 presents the proposed scheduler, Section 4 provides testing details, the discussion is covered in Sections 5 and 6 concludes.

2. MES and AIVs

History of MES use starts in the 1970s when the manufacturing industry started using software applications. Initially, these applications were used to provide basic inventory management. After some time, the manufacturing industry realized the need for software for the effective management of processes involved in manufacturing. In the early 1980s, more advanced software applications came with the introduction of MRP (Manufacturing Resource Planning). The MRP mainly improved the resource/material management but this was not suitable for the real-time control and management of the shop floor operations. Therefore, the industries started looking towards Enterprise Resource Planning (ERP) applications that were intended to integrate the shop floor functions for better forecasting, planning, purchase, costing, accounting, and inventory management functionalities. However, still there was a need for the data collection software to facilitate real-time controlling/management. During the 1980s, many applications were designed for operations, such as scheduling, data collection, and maintenance. In the early 1990s, these applications were transformed into what we now know as the MES. Initial models of the MES were developed as on-site applications and represented the manufacturing process as-is. These systems were rigid, and it was very difficult to implement changes, as for changes many efforts were needed in terms of coding. In the middle of the 2000s, there was a transition of MES from rigid systems to highly flexible and off-site web-based applications. As a result, MESs became more flexible, modular and off-site web-based applications became popular. These applications could be used from anywhere in the world for any plant through the use of an internet connection. The modern MES applications handle real-time data from every aspect of the manufacturing process.

With the advent of Industry 4.0 the MES will play an integral role. The MES will be the central information and data hub for production and all areas involved throughout the industry. Industry 4.0 also aims to improve productivity by fulfilling the growing customer demands for a faster real-time response via decentralized production control. These expectations can be fulfilled by the MES to improve performance, quality, and agility for globalized manufacturing businesses. For smart manufacturing, the MES is one of the essential enablers, a stepping stone on the path to facilitate
Industry 4.0 that all manufacturers must introduce to succeed in today’s marketplace. Here, the SAP MES [32] is used, and it is accessed using LabVIEW software through webservices.

MES integration with an AIV is an important element to develop a smart manufacturing environment. For example, in a smart factory, if a product has been passed from all processing steps and MES shows that this product is ready to pick, then an AIV can be directed to pick the product based on the MES data and drop it at a specified location. The drop-off location can be determined based on the information receiving from MES. However, all this is possible only when there will be a communication link between the MES and the AIV.

The term AIV refers to robotic vehicles that can navigate without human intervention [33]. Today, a modern manufacturing business can hardly imagine working without the involvement of AIVs. The integration of AIVs with smart technologies allows the creation of more independent robotic systems. These systems are not only able to carry out basic repetitive operations, such as assembling, loading, or modifying parts, but can also perform cognitive tasks, improve processes without human intervention and take instant decisions. AIV technology is growing in the manufacturing industry and can advance automotive manufacturing by complementing the flexibility of material handling in the production line. Here, the focus is on AIVs built using ROS [34–36] that provide basic level control but do not provide advanced features like fleet management and integration with MES. These features can be developed for low-cost AIVs using a LabVIEW-based scheduler. Here, the focus is to provide a solution to integrate these AIVs with the MES. Using the proposed scheduler these low-cost AIVs can be used to develop an automated, fully connected customized solution to build a smart manufacturing environment, which is the ultimate goal of Industry 4.0.

ROS is a meta-operating system for robots [37]. As mentioned in Section 1, this is an open-source platform which functions are equivalent to what we expect from an operating system. These functions include low-level device control, hardware abstraction, message-passing between processes, implementation of commonly-used functionality, and package management. The ROS meta-operating system also provides libraries and tools for obtaining, building, writing, and running code across multiple computers. Python and C++ are mostly used as the ROS programming languages. ROS provides peer to peer communication. The greatest benefit of using ROS as an operating platform is the ability to reuse code and/or build upon code already available in library banks of the ROS network. In the ROS environment, nodes are used to send and receive data. Nodes that send data are called Publisher nodes, while nodes that receive data are called Subscriber nodes. These nodes register as Publisher or Subscriber to a program called ‘ROS Master’. This is necessary because Publisher and Subscriber nodes do not communicate directly with each other but through the ‘ROS Master’ [37]. The communication is simplified by publishing or subscribing messages to ROS topics. ROS related terms like node, topic, message, ROS Master, etc., are explained in Reference [37].

The next section provides details about the proposed Scheduler development.

3. Proposed Scheduler to integrate the MES and AIV

As mentioned in Section 1, the proposed scheduler provides integration of the MES and AIV. This scheduler is developed using LabVIEW software and acts as a middleware between the MES and an AIV. This section is divided into three parts as follows and each part is explained below.

- LabVIEW-MES communication.
- LabVIEW-AIV communication.
- Scheduler development.

3.1. LabVIEW-MES Communication

This section provides details about how LabVIEW communicates with MES webservices. As mentioned before, the webservice development is out of the scope of this paper and the webservices needed for this project are developed and provided by the industrial partner. For this section,
an example of a single MES webservice named ‘WebserviceA’ is considered, that provides details about the status of pickup locations. The same method is valid for all the webservices mentioned in Section 3.3.

Firstly, the mentioned webservice is imported into LabVIEW using the WSDL (WebService Description Language) link. The WSDL link describes information about the specific webservice. This information includes the available operations (functions or methods) and data types. The successful import operation of webservice loaded the equivalent LabVIEW VI (Virtual Instrument) along with some additional VIs, which are necessary to build a communication link with the MES. Most parameters of the imported webservice are not simple values but are references to the .NET objects, this is shown in the front panel of VI in Figure 4. This VI cannot be used as-is because we need the request and response objects to communicate with the MES. So, the VI is edited using .NET functions in LabVIEW to create the request and response .NET objects and to set properties or to call methods on those objects. The block diagram to add request and response objects is given in Figure 5 and explained below.

![Figure 4. Front panel of 'WebserviceA' webservice (before edits).](image-url)

![Figure 5. Block diagram to edit LabVIEW VI of imported webservices.](image-url)

To create the request object for the selected webservice ‘WebserviceA’, the constructor node is used with the object ‘WebserviceAREQ’. This constructor node is connected to the property node, which displays the request parameters of the webservice. Here, the request parameters are ‘operation’ and ‘resourcelid’. The ‘Operation’ input can be configured with PICK (for PICK operation) or DROP
(for DROPOperation). The resourceId is a virtual representation of the physical location so it can be configured with PICK1 to represent pickup location 1 or DROP1 to represent drop-off location 1. The resourceId is selected based on the configured operation. The scheduler supports a number of pickup and drop-off locations (for details, see Section 3.3). Details about the selection of pickup and drop-off locations are given in Sections 3.3.2 and 3.3.6, respectively.

Another constructor node is created with the 'WebserviceA' object, which connects with the invoke node through the property node, to invoke a method called 'WebserviceA'. The invoke node connects with open and close webservice VIs. The invoke node is connected to a new property node to return the response of 'WebserviceA' method. This response is read by the property node of 'WebserviceARSP' object. The response terminals of this property node are connected to LabVIEW indicators, while 'FOR Loop' and 'Nested Loop' are used to extract the response. The number of 'FOR loop' depends on the number of pickup or drop-off locations. Similarly, the number of 'Nested loop' depends on the number of products at the relevant pickup or drop-off locations. In the case of multiple loops, all 'FOR Loop' will be connected to resourcelist, and all 'Nested Loop' will be connected with sfclist (as shown in Figure 5) but the difference will be the Loop count. The information about the maximum number of pickup and drop-off locations and the maximum number of product support at each location should be known in advance to develop the scheduler accordingly.

The edited VI front panel of the 'WebserviceA' webservice is shown in Figure 6, where details of the product ready are shown under sfclist. In Figure 6, the resourceId response represents the location where the product is placed and resourceStatus response represents the status of that location. If there is no product to pick at PICK1, then sfclist fields will be empty. The front panel of the edited VI shows the request and response options that are sent to and received from the MES, respectively. Based on the response of Figure 6, LabVIEW can send the command to an AIV to go to PICK1 location and pick the available product. In the case of multiple pickups or drop-offs locations, the selection can be made based upon priority rules. As mentioned earlier, in the same way, all the webservies, discussed in Section 3.3, are imported in LabVIEW, and equivalent VIs are edited and configured accordingly.

![Figure 6. Front panel of 'WebserviceA' webservice (after edits) (sfclist data is not displayed because of its specific company product details).](image)

The next section provides LabVIEW and AIV communication details.

3.2. LabVIEW-AIV Communication

As mentioned earlier, the ROS-built AIVs are used in this project. This section involves the development of two software packages, namely: (a) ROS package and (b) LabVIEW Application. Details of these two software packages are given below.

3.2.1. ROS Package

As mentioned in Section 2, ROS exchanges data using Publisher and Subscriber nodes. Here, the ROS package is developed using C++. In the ROS package, a Subscriber is generated which,
subscribes to a topic (topic published by LabVIEW Publisher) to receive commands for the AIV from LabVIEW. Similarly, a Publisher is generated which publishes a topic (topic subscribed by LabVIEW Subscriber) to send an acknowledgment to LabVIEW. To take coordinates of the physical location, a map is generated using SLAM (Simultaneous Localization And Mapping) [37] technique. To generate the map, the AIV is taken round the workplace and allowed to scan the surrounding area with its main LiDAR sensor. Here, hard coded coordinates of physical locations are used. The coordinates are taken using amcl (adaptive Monte Carlo Localization) node [37], which takes laser scans, transform messages, and outputs estimated position. The pseudo-code for the ROS Publisher/Subscriber used here is given in Figure 7. This code is compiled and executed using the respective ROS commands [37].

```
BEGIN:
  Include Headers;
  Input string s;
  Assign coordinate values to x, y, z, w constants;
  Declare Callback function to call when receive message;
  Declare Movetogoal (x, y, z, w) function;
Main:
  ROS node initialization;
  Start the Node;
  Declare a Publisher;
  Declare a Subscriber;
  Specify loop frequency;
  Call Subscriber using Ros::SpinOnce function and save received string in ‘s’ variable;
  While (a = exit)
  { If (s = 'Go_to_Location_PA')
    { Movetogoal (A.x, A.y, A.z, A.w);
      If (goalreached)
      { Publish message 'Reached_Location_PA';
        else {Publish message 'Location_PA_Unreachable';
      }
    }
  else if (s = 'Go_to_Location_PB')
  { Movetogoal (B.x, B.y, B.z, B.w);
      If (goalreached)
      { Publish message 'Reached_Location_PB';
        else {Publish message 'Location_PB_Unreachable';
      }
    }
  ……….
}  END
```

Figure 7. Pseudo-code for Robot Operating System (ROS) Publisher/Subscriber node.

The ‘Movetogoal’ function of Figure 7 defines a client, which is responsible for sending navigation goal requests to the move_base node. The move_base node sends the goal to the global path planner, which finds a global path from the robot location to the goal location. The global path planner calculates the safe path to reach the goal location and also sends this trajectory to the local planner. The local planner then executes each segment of the global plan. So, given a plan to follow and a map, the local planner provides velocity commands to move the AIV. The local planner also monitors the odometry and laser data and selects a collision-free local plan for the AIV. The local planner recomputes the path in real-time to keep the AIV safe from striking objects, yet still allowing the AIV to reach the destination. Details about the global and local path planner and obstacle avoidance are given in Reference [37]. According to Figure 7, if strings ‘Go_to_Location_PA’ or ‘Go_to_Location_PB’ are received from LabVIEW, then AIV will go to the physical location PA or PB, respectively. The coordinates of PA and PB locations are defined as A.x, A.y, A.z, A.w and B.x, B.y, B.z, B.w, respectively. In Figure 7, only two locations (PA
and PB) are shown as an example, while in this work total six locations are defined (three pickups and three drop-offs). Note that, here and in Section 3.3, PA, PB, and PC are used for pickup locations and represent PICK1, PICK2, and PICK3 (mentioned in Section 3.1) respectively. Similarly, DA, DB, and DC are used for drop-off locations and represent DROP1, DROP2, and DROP3, respectively.

3.2.2. LabVIEW Application

As mentioned in Section 1, the LabVIEW add-on tool ‘ROS for LabVIEW’ is used here. This LabVIEW tool is used to communicate with ROS applications and can handle node-to-node transport negotiation and communication setup using TCP as its transport mechanism.

The LabVIEW Publisher and Subscriber continuously run using a ‘While Loop’ to send and receive messages to/from ROS applications running on the AIV. The ‘ROS for LabVIEW’ tool includes a set of VIs that establish a connection with ROS applications. Some of these VIs are used to design the Publisher and Subscriber node as shown in Figure 8. Each of these VIs performs specific tasks, as explained below.

**Figure 8.** Block diagram of LabVIEW Publisher/Subscriber node. (a) Publisher; (b) Subscriber.

The ROS node is initialized using ROS_Topic_init VI, which takes the following as inputs (a) topic name, (b) topic type, (c) action (Publisher or Subscriber), (d) update rate, and (e) queue size. The node name (/LV1) is also assigned using the ‘node’ terminal of the mentioned VI. The ROS_Topic_init VI is connected to ROS_Topic_Read VI (in case of Subscriber) and ROS_Topic_Write VI (in case of Publisher). The ROS_Topic_Read VI is used to receive messages and ROS_Topic_Write VI is used to send messages. The messages are sent to the AIV in the form of strings. The Subscriber and Publisher topics are closed using ROS_Topic_Close VI.
3.3. Scheduler Development

The scheduler is designed to transport products from one location to another (in a manufacturing environment) through an AIV based on MES demand. The scheduler is configurable and designed to currently support the maximum of three pickups and three drop-offs locations (as mentioned earlier). Each pickup location shows one active product at a time ready for pick while the number of products at drop-off locations is configurable (can be set using scheduler interface). Here, the scheduler is explained using the maximum of four products at each drop-off location. The scheduler processing cycle starts by looking for products available to pick at the selected pickup location and the cycle ends after dropping the picked product at the selected drop-off location. This one complete cycle consists of 8 steps as shown in Figure 9. The flat sequence structure of LabVIEW is used in a ‘while Loop’ to implement these steps sequentially and continuously. In these steps, the scheduler communicates with the MES and AIV in the same way as mentioned in Sections 3.1 and 3.2, respectively. The steps mentioned in Figure 9 are explained below.

![Figure 9. Steps of one complete cycle of scheduler operation.](image-url)

3.3.1. Check Status of Pickup Locations

Firstly, ‘WebserviceA’ of the MES is imported into LabVIEW, and then an equivalent VI is edited and configured accordingly, as discussed in Section 3.1. This webservice provides product availability information at pickup locations. The operation of this step is shown in Figure 10. The scheduler sends a request to the MES, this request is in terms of operation and resourceId fields. Here, the PICK operation is requested, and as a response the scheduler receives many parameters from the MES, as shown in Figure 10. The information about the total number of active pickup locations (supported by MES) and the number of active products at each pickup location is not directly provided from the MES. This information is extracted by applying the ‘Array Size’ function of LabVIEW on resourceList (for pickup locations) and sfcList (for active products) response. The scheduler continuously looks for products available for pick. If there is no product at any pickup location, the scheduler will send the AIV to a temporary holding location (PICK_Holding_Bay) and the scheduler will continuously keep looking for active products again. The PICK_Holding_Bay is a predefined location, which is relatively near to the pickup locations. This feature was introduced to save AIV travel time as the AIV can reach selected pickup locations faster. If the product will be available at any pickup location, then the scheduler will transfer product details to the next step.
Figure 10. Check status of pickup locations.
3.3.2. Pickup Location Selection

As mentioned earlier, there are three pickup locations. So, in case of products available at all the pickup locations simultaneously, the scheduler needs to select one location based on the sequential logic presented in this step. The sequential logic is build using a counter. A counter ‘Cp’ (Counter Pick) is initialized which can count a maximum of three in a repetitive sequence of 1,2,3—1,2,3—1,2,3 and so on. Typically, this Cp is updated one time in each cycle of operation. Although there are some other cases mentioned below when Cp will be updated within the operation of this step. Here, three products (sfc-A, sfc-B and sfc-C) are received from step-1. There are three possible cases here. The first case is: when Cp = ‘1’ and sfc-A is available, the second case is: when Cp = ‘2’ and sfc-B is available, and the third case is: when Cp = ‘3’ and sfc-C is available, other cases will not be responded to. For example, if Cp = ‘1’ and only sfc-B is available, then Cp will be updated, and this will result in the above mentioned second case. Details of the first case are given below, and the same is valid for the remaining two cases.

If count = ‘1’ and sfc-A is available, then the scheduler will select Location_PA at pick and will send an AIV to this selected location. The scheduler will wait for the response from the AIV and will move towards step-3 if the response ‘ Reached_Location_PA’ is received. Otherwise, in the case of ‘ Location_PA_Unreachable’ response (e.g., due to unavoidable obstacle), the Cp will be updated, and the scheduler will check the next case from above mentioned three cases. The Cp will be updated in one more case, i.e., count = ‘1’ and sfc-A not available.

An optional feature of a timer is also introduced for all mentioned three cases. The timer is introduced to deal with a situation when the scheduler is waiting for a reached or an unreached response from the AIV, and there is no response (because of an unexpected reason) within the defined time limit, i.e., ‘Target_Time’ (which can be set using scheduler interface). In this case, the AIV needs to be manually reset. The operation of this step is shown in Figure 11.

3.3.3. Pick Product

The operation of this step is shown in Figure 12. Selected pickup locations of Section 3.3.2 are named as ‘Pre-Dock’ locations. After reaching the pre-dock location, the AIV will switch into the ‘Auto-Docking’ mode. After successful docking at the pickup location, the AIV will communicate with the lift system and will pick the product (development of lift system is out of the scope of this work). The scheduler will wait for the response from the AIV. One of the following three responses is expected: PICK_Operation_Successful, PICK_Operation_Failed, or PICK: No_Responce_from_Lift.

PICK_Operation_Successful: This response means the product has been picked successfully. In this case, the scheduler will forward this signal to the next step.

PICK_Operation_Failed: This response means it is not possible to pick the product (because pre-docking is failed). In this case, the scheduler will send the AIV to a predefined ‘Error Zone’ and manual intervention is needed.

PICK: No_Responce_from_Lift: This response means AIV cannot establish communication with the lift system. In this case, the scheduler will send the AIV to a predefined ‘Error Zone’ (same as mentioned in the case of PICK_Operation_Failed) and manual intervention is needed.
The scheduler will check the next case from the above mentioned three cases. The \( C_p \) will be updated in one more case, i.e., \( count = '1' \) and \( sfc-A \) is not available.

An optional feature of a timer is also introduced for all mentioned three cases. The timer is introduced to deal with a situation when the scheduler is waiting for a reached or an unreached response from the AIV, and there is no response (because of an unexpected reason) within the defined time limit, i.e., \( Target\_Time \) (which can be set using the scheduler interface). In this case, the AIV needs to be manually reset. The operation of this step is shown in Figure 11.

Figure 11. Pickup location selection.
3.3.4. Send Product-Pick Confirmation to MES

This involves communication between the scheduler and the MES through another webservice ‘WebserviceB’. This webservice is imported in LabVIEW, and the equivalent VI is edited and configured accordingly, as mentioned in Section 3.1. The operation of this step is shown in Figure 13. This webservice is used to confirm to the MES about the successful PICK operation so that the MES can remove the entry of the relevant product (which has been picked) at the pick. After the successful product-pick operation (mentioned in Section 3.3.3), the scheduler sends a request to the MES (through ‘WebserviceB’) based on the response of Boolean ‘AND’ logic, as shown in Figure 13. At a time, the output of one ‘AND’ logic will be high. Here, three parameters, i.e., operation, resourceId, and sfc (product Id), are applied as a request to the MES. The sfc is the Id of product, which has been picked and the requested resourceId represents the selected pickup location of Section 3.3.2. The MES will return an error in case of the wrong resourceId applied as a request. After the successful removal of product entry, the MES will send a response to the scheduler. One of the responses of the MES is ‘NextOperation’ (shown in Figure 13), which is checked by the scheduler, and, if it is equal to ‘DROP’, then the scheduler will proceed towards the next step. In addition, on the successful completion of this step, the scheduler will update the number of products at pickup locations on the interface. An optional feature of the timer is also added in this step. This is useful if there is no response from the MES within the ‘target_time’ limit (which can be set using the scheduler interface). In the case of time_out, an operator needs to investigate the issue by checking (a) Schedular-MES connection and(or) (b) By checking the ‘WebserviceB’ operation. This is the last step of PICK operation, and the next four steps are of DROP operation.
3.3.5. Check Status of Drop-off Locations

For this step, the MES webservice ‘WebServiceC’ is imported in LabVIEW. This webservice provides the status of drop-off locations, which includes the number of active drop-off locations, the number of products available at each drop-off location, and details of available products. The request and response of the MES are shown in Figure 14. Like Section 3.3.1, the information of the number of products and the number of drop-off locations is extracted by applying the ‘Array Size’ function of LabVIEW on resourceList and sfcList response. Each drop-off location has support for the maximum four products in Figure 14 (this is configurable through the scheduler interface). The purpose of this step is to check the status of drop-off locations and after this, the scheduler will proceed to step 6.
3.3.6. Drop-off Location Selection

In this step, a drop-off location is selected sequentially based on the information received from the MES. The operation of this step is shown in Figure 15. A counter ‘Cd’ (Counter DROP) is initialized that can count to a maximum of three in a repetitive sequence (same as Cp). Like Cp, this Cd is normally updated in each cycle of operation (some cases mentioned here will also update Cd within the operation of this step). The scheduler is designed for three drop-off locations and the selection criteria can be built for any condition using programming in LabVIEW. Here, the drop-off locations with the following two conditions will not be selected. (a) Drop-off locations with four products; and (b) Drop-off locations with the resourceStatus equal to ‘Unscheduled Down’. The scheduler will check conditions mentioned in Figure 15 using Boolean ‘AND’ logic to select a drop-off location. If the outcome of a particular condition (mentioned in Figure 15) is false, then Cd will be updated, and the scheduler will check the next condition using the same Boolean ‘AND’ logic. After the drop-off location selection, the AIV will receive a signal from the scheduler to go to the selected drop-off location. In the case where all drop-off locations have four products, the AIV will be directed to go to the DROP_Holding_Bay location. The ‘DROP_Holding_Bay’ is a predefined location that is near the drop-off locations. An optional feature of the timer is also introduced here (same as the timer mentioned in Section 3.3.2). The step-5 is instantiated to continuously check the status of the number of products at drop-off locations within this step of the operation.

3.3.7. Drop Product

This step is the same as mentioned in Section 3.3.3. The only difference is that the product will be dropped at the selected drop-off location of Section 3.3.6. This step function is summarized in Figure 16.

3.3.8. Send Product-Drop Confirmation to MES

This step involves communication between the scheduler and ‘WebserviceD’ webservice of MES. Like previous steps, this webservice is also imported in LabVIEW, and its equivalent VI is edited and configured accordingly, as mentioned in Section 3.1. The operation of this step is shown in Figure 17. This webservice is used to send the confirmation signal to the MES about the successful product-drop operation, so that the MES can add the entry of relevant product (which has been dropped) at the drop. The scheduler sends a request to the MES (through ‘WebserviceD’) based on different conditions, which are checked by Boolean ‘AND’ logic, as shown in Figure 17. In fact, there is a total of nine conditions out of which three are mentioned here. Remember that here only one product will be ready at a time for operation and this is the product, which product-pick confirmation (given in Section 3.3.4) has been received by the MES. The response of the MES for each case is also shown in Figure 17. After the
successful addition of requested product entry at a drop-off location, the MES will send the response to the scheduler. The scheduler will check the 'resultText' field of the MES responses. If this field response is equal to 'Successful', then the scheduler will update the number of products at drop-off locations, and this is the completion of one cycle of the scheduler. Now, the next cycle will again start from step-1 (Section 3.3.1). An optional feature of the timer is also added in this step, in which function is the same as the timer function mentioned in Section 3.3.4.

**Figure 15.** Check status of drop-off locations.

**Figure 16.** Drop product.
In case of low battery, AIV sends 'low battery' signal to the scheduler. The scheduler checks battery status of AIV before start of the above-mentioned step-1 of Figure 9, in each cycle of operation. The scheduler also supports data logging of all steps of scheduler operation.

Although the scheduler development explained here was for a specific industrial scenario, the authors believe that this work provided a strong foundation to develop sustainable, high performing customized solutions to fulfil the needs of an industry that want to integrate an MES and AIVs.

The next section provides testing details of the scheduler presented here.

4. Testing

The proposed scheduler can be used to integrate any ROS-built AIV with an MES. This scheduler testing is performed with a low-cost AIV and an MES system. During the scheduler development, the Turtlebot-3 burger model [14] is used to test LabVIEW-ROS communication. The Turtlebot-3 is a
good research platform to explore about ROS. While final testing is performed on a low-cost Robotnik AIV ‘RB-1’. The Turtlebot-3 and RB-1 are shown in Figures 3 and 18 accordingly. The ‘RB-1’ is a development robot and it is low-cost because it is purely a research-oriented platform and does not provide advanced features mentioned in Section 2. The scheduler-MES communication is tested using a SAP MES system. A remote system (Windows-based) is used to run the scheduler application. The remote system is connected to RB-1 on which ubuntu 16.04 and ROS (Kinetic) is installed. The network is configured in such a way that the remote system and RB-1 should be on the same network. Initially, LabVIEW-MES and LabVIEW-AIV communication was tested separately, and then the overall system communications was tested.

![Turtlebot-3](image)

**Figure 18.** Turtlebot-3.

As mentioned in Section 3.2.1, firstly a map of target testing area is generated using the ‘RB-1’, and coordinates of the selected physical locations are noted. The ROS package is prepared and executed as mentioned in Section 3.2.1, and then ROS navigation is launched. With ROS navigation, the visualization tool RViz is also executed. The pose of the robot is estimated using RViz tool.

After setting up the ‘RB-1’, the scheduler is executed and it pops up a window at start-up, for the IP address of the ROS Master (running on RB-1). Using this IP address, the scheduler establishes a connection with RB-1 through a TCP connection. If there is any issue with the connection, the same window will pop up again. The scheduler provides options to enable/disable pickup and drop-off locations, timers function, and Holding_Bay locations support. Each event is indicated through flags; for example, selected location, RB-1 reached selected location or not, timeout and current status of scheduler, are all indicated in real-time using flag indicators. The target_time and maximum number of products at drop-off locations can also be set using the scheduler interface. The scheduler also shows the commands sends from the scheduler to the AIV and acknowledgments from the AIV to the scheduler.

The timing details of the scheduler is taken from the log file. The log file records the execution time of each step given in Table 1. The maximum time for the pick operation is taken by step-2 and step-3, which involves preparing the robot for docking operation and to reach the pre-doc location at the pick point. Similarly, the maximum time for the drop operation is taken by step-6 and step-7 for docking preparation and to reach the pre-doc location at the drop point. The testing setup with pickup, drop-off, error zone, and holding bay locations is shown in Figure 19. The time period mentioned in the table is taken using the setup mentioned in Figure 19, and this operation does not involve holding_bay or error zone selection. There is literature published on the scheduler presented in Reference [38–41], but, to the best of authors’ knowledge, there is no published literature on the timing details of the same scheduler which provides an integration of a MES and AIV. So, a fair comparison of timing is not possible.
Figure 19. Testing setup. Here, the coordinates of single pickup location are used for two pickup locations and similarly scheduler is tested for three drop-off locations by using coordinates of single drop-off location.

The scheduler interface is shown in Figure 20. This scheduler shows two pickup locations (although the develop scheduler can support maximum 03 pickup locations, as shown in Section 3.3, but the MES was configured to support only two pickup locations, so the third pickup location is not shown on front-panel) and three drop-off locations.

Figure 20. Scheduler interface

5. Discussion

There is no out of the box solutions to connect an SAP MES with a ROS-based AIV that the authors are aware of. Different organizations and consortiums are working to come up with a solution. Recently, ROS-Industrial had the opportunity to collaborate with Microsoft, BMW, and Open Robotics on an automation solution to integrate SAP with fleet management and navigation functionality [42], as BMW required a way to connect their robots to an open-source platform, allowing a heterogeneous fleet of robots to work together in perfect harmony with the human workers on the assembly line.

The main objective of this work was to provide a possible solution to integrate a ROS-built low-cost AIVs with a MES. The literature review shows that there is published work available on MES related research and robotic automation but to the best of the authors’ knowledge, no published work provides a detailed solution on how to integrate a low-cost AIVs with an MES. As mentioned in Section 1, there are some AIVs [30,31] commercially available, which offer communication with MES but the manufacturers of these AIVs offer industrial solutions at a very high-cost and under
license. In addition, the greatest benefit to a company to building customized solutions using low-cost, self-build AIVs is the ownership of products and systems and potential sales to partner companies.

The proposed integration solution here used ROS-built AIVs because ROS is open source and does not have any license requirements. The scheduler is developed using LabVIEW software, and as mentioned earlier, the main reason to select LabVIEW as a middleware platform is that it has support for both webservice and the ROS platform. The LabVIEW platform is not new for industries, it is used for many applications like building maintenance, testing, monitoring, and automation applications. LabVIEW is used in this work because it is an attractive platform for users with a non-programming background and comparatively quick programming is possible using LabVIEW. Other software tools can require programming abilities or have limitations with regards to the integration of the AIV and MES. For example, Node-RED has support for webservice but ROS support is limited [43]. In addition, LabVIEW has support for the widely used industrial protocols like OPC UA (Open Platform Communications Unified Architecture), MQTT (Message Queue Telemetry Transport), and AMQP (Advanced Message Queuing Protocol) [44–46]. A brief discussion about these industrial protocols is given below.

Industrial protocols are real-time communication protocols, developed to interconnect the systems, interfaces, and instruments. The main goal of industrial protocols is to apply one communication technology across all levels of factory automation. The OPC UA [47,48] is currently the most widely used industrial protocol. This protocol supports high scalability, availability, and implements Internet capabilities, as well as secure cross-platform data exchange. The new features of OPC enable a model in which a server sends its data to the network (publish) and every client can receive this data (subscribe). The MQTT [49] is another industrial protocol that was created to collect data from many devices and then transport that data to the IT infrastructure. It is lightweight and, therefore, ideal for remote monitoring. The publisher/subscriber model of MQTT decouples the client that sends a message (publisher) from the client or clients that receive the messages (subscribers). One more example of the industrial protocol is AMQP [50], which creates interoperability between clients and brokers. Its goal of creation was to enable a wide range of different applications and systems to be able to work together, regardless of their internal designs, standardizing enterprise messaging on an industrial scale.

The work presented here is designed for a cell-based manufacturing environment. The traditional linear manufacturing production lines are not flexible and are generally designed for one product type. Development of the new smart manufacturing assembly lines, i.e., cell-based, allows flexibility of products during volume fluctuations. The smart manufacturing assembly lines are used for multiple products, having the capability of a flexible production system for all production operations. In cell-based manufacturing, different cells are connected by an automatic product delivery system that transports the product through the manufacturing process, from cell to cell as defined by detailed production sequences using an MES. The automatic product delivery system on the AIV can collect and deliver products to process cells in a format suitable for product feed.

The proposed scheduler currently has support for a single AIV, but this scheduler can be updated to support a fleet of AIVs in a single manufacturing place. A detailed review of swarm robotics system and software for multi-robotic environment are given in Reference [51,52], respectively. Many AIVs or AMRs (Autonomous Mobile Robots) available in the market with Fleet management already take care of scheduling but either these are very expensive and/or require expensive licenses/contracts for continuous technical support [53]. For industries that do not wish to use this expensive solution or want a bespoke solution, can take advantage of this work. The aim of this research was to provide a cost effective and customized solution for the industry partner to build an integrated and automated industrial environment which is a requirement for Industry 4.0. Here, a separate customized scheduler is discussed but the authors believe that this provides a platform to build a low cost fully integrated fleet manager with the scheduling feature integrated.

The proposed scheduler is highly configurable and can be used to integrate any ROS-built AIV with an MES. The scheduler interface is user friendly and can be operated by a non-technical operator.
As mentioned in Sections 3.3.2 and 3.3.6, the selection criteria for pickup and drop-off locations selection is based on sequential logic. But the scheduler can be designed to support more selection criteria, like the selection of nearest pickup and drop-off locations, selection of the high priority locations/product, etc.

The development of an increase in automation technologies will drive the creation of more profitable and efficient manufacturing. This is a vital requirement for the coming years, with increased pressure on production in high-cost countries. This research work will have a direct influence on the future product direction with low-cost AIVs and will potentially increase production at the plant and thus increased jobs.

6. Conclusions

Industry 4.0 is the current revolution in the manufacturing industry which involves the integration of sensors, production machines, wireless connectivity, and links these to a software platform that can oversee the whole production line process and execute decisions autonomously. Interconnectivity and automation are the key factors of Industry 4.0 and are needed to implement a smart manufacturing environment. This work provides a solution to integrate a low-cost ROS-based AIVs with an MES. The ROS-built AIVs are targeted because ROS is open source, free license, and a large ROS library is available. The proposed solution used a LabVIEW-based scheduler as a middleware between the AIV and the MES. The proposed scheduler is configurable and provides options to enable/disable available features and offers error handling in the case of some specific scenarios that the industry partner required. The scheduler controls the AIVs to collect and deliver products to process cells, based on the demand of an MES in a cell-based manufacturing environment. This proposed integration will provide a boost to productivity, flexibility, and quality in a smart manufacturing environment at a low cost.

Author Contributions: Conceptualization, M.R. and T.N.; funding acquisition, T.N.; investigation, M.R., L.L. and J.C.; methodology, M.R.; project administration, T.N. and D.T.; resources, M.R. and L.L.; software, M.R.; supervision, T.N.; writing—original draft, M.R.; writing—review and editing, M.R., T.N. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by following: Science Foundation Ireland Centres: Lero, the Irish Software Research Centre, Grant No 13/RC/2094 and Confirm, the Smart Manufacturing Centre, Grant No 16/RC/3918 and co-funded under the European Regional Development Fund through the Southern & Eastern Regional Operational Programme.

Acknowledgments: Authors would like to thank their industrial partner, Kostal Ireland, and academic partner IT Tralee, Ireland for collaboration on this project.

Conflicts of Interest: The authors declare no conflict of interest.

References
1. Tay, S.; Chuan, L.; Aziati, A.; Ahmed, A. An overview of industry 4.0: Definition, components, and government initiatives. J. Adv. Res. Dyn. Control Syst. 2018, 10, 1379–1387.
2. Vaidya, S.; Ambad, P.; Bhosle, S. Industry 4.0—A Glimpse. Procedia Manuf. 2018, 20, 233–238. [CrossRef]
3. Liao, Y.; Deschamps, F.; Loures, E.; Ramos, L. Past, present and future of Industry 4.0—A systematic literature review and research agenda proposal. Int. J. Prod. Res. 2017, 55, 3609–3629. [CrossRef]
4. Crafts, N. The First Industrial Revolution: Resolving the Slow Growth/Rapid Industrialization Paradox. J. Eur. Econ. Assoc. 2005, 3, 525–534. [CrossRef]
5. Huberman, M.; Meissner, C.; Oosterlinck, K. Technology and Geography in the Second Industrial Revolution: New Evidence from the Margins of Trade. J. Econ. Hist. 2017, 77, 39–89. [CrossRef]
6. Taalbi, J. Origins and pathways of innovation in the third industrial revolution1. Ind. Corp. Chang. 2018. [CrossRef]
7. Veil, J.; Kiel, D.; Müller, J.; Voigt, K. Lessons learned from Industry 4.0 implementation in the German manufacturing industry. J. Manuf. Technol. Manag. 2019. [CrossRef]
8. Alcacer, V.; Machado, V. Scanning the Industry 4.0: A Literature Review on Technologies for Manufacturing Systems. Eng. Sci. Technol. Int. J. 2019, 22, 899–919. [CrossRef]
9. Oztemel, E.; Gursey, S. Literature review of Industry 4.0 and related technologies. *J. Intell. Manuf.* 2020, 31, 127–182. [CrossRef]
10. A Rojko Industry 4.0 Concept: Background and Overview. *Int. J. Interact. Mob. Technol.* 2017, 11, 77. [CrossRef]
11. Vladimir, V.; Vasilev, P. Manufacturing Operations Management—The Smart Backbone of Industry 4.0. *Industry 4.0* 2016, 1, 71–76, ISSN 1310-3946.
12. Mantravadi, S.; Møller, C. An Overview of Next-generation Manufacturing Execution Systems: How important is MES for Industry 4.0? *Proc. Manuf.* 2019, 30, 588–595. [CrossRef]
13. Melanson, T. What Industry 4.0 Means for Manufacturers. Blog. Available online: https://aethon.com/mobile-robots-and-industry4-0/ (accessed on 20 April 2020).
14. Turtlebot-3. Available online: https://www.turtlebot.com/ (accessed on 20 April 2020).
15. Robotnik—RB-1. Available online: https://www.robotnik.eu/mobile-robots/rb-1-base-2/ (accessed on 20 April 2020).
16. Bahrin, M.; Azli, M.F.; Talib, M. INDUSTRY 4.0: A review on industrial Automation and robotic. *J. Teknol.* 2016, 78, 6–13. [CrossRef]
17. Pfeiffer, S. Robots, Industry 4.0 and Humans, or Why Assembly Work Is More than Routine Work. *Societies* 2016, 6, 16. [CrossRef]
18. Aalst, W.; Bichler, M.; Heinzl, A. Robotic Process Automation. *Bus. Inf. Syst. Eng.* 2018, 60, 269–272. [CrossRef]
19. Hellmann, W.; Marino, D.; Megahed, M.; Suggs, M.; Borowski, J.; Negahban, A. Human, AGV or AIV? An integrated framework for material handling system selection with real-world application in an injection molding facility. *Int. J. Adv. Manuf. Technol.* 2019, 101, 815–824. [CrossRef]
20. Tokody, D.; Mezei, I.; Schuster, G. An Overview of Autonomous Intelligent Vehicle Systems. In *Vehicle and Automotive Engineering. Lecture Notes in Mechanical Engineering*; Jármai, K., Bolló, B., Eds.; Springer: Cham, Switzerland, 2017; pp. 287–307. [CrossRef]
21. Ugarte, B.; Artiba, A.; Pellerin, R. Manufacturing execution system—A literature review. *Prod. Plan. Control* 2009, 20, 525–539. [CrossRef]
22. Zhong, R.; Dai, Q.; Qu, T.; Hu, G.; Huang, G. RFID-enabled real-time manufacturing execution system for mass-customization production. *Robot. Comput. Integr. Manuf.* 2013, 29, 283–292. [CrossRef]
23. Helo, P.; Suorsa, M.; Hao, Y.; Anussornitisarn, P. Toward a cloud-based manufacturing execution system for distributed manufacturing. *Comput. Ind.* 2014, 65, 646–656. [CrossRef]
24. Bratukhin, A.; Sauter, T. Functional Analysis of Manufacturing Execution System Distribution. *IEEE Trans. Ind. Inform.* 2011, 7, 740–749. [CrossRef]
25. Arica, E.; Powell, D. Status and Future of Manufacturing Execution Systems. In Proceedings of the 2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), Singapore, 10–13 December 2017; pp. 2000–2004. [CrossRef]
26. Lynch, L.; McGuinness, F.; Clifford, J.; Rao, M.; Walsh, J.; Toal, D.; Newe, T. Integration of autonomous intelligent vehicles into manufacturing environments: Challenges. *Proc. Manuf.* 2019, 38, 1683–1690. [CrossRef]
27. Cronin, C.; Conway, A.; Walsh, J. State-of-the-Art Review of Autonomous Intelligent Vehicles (AIV) Technologies for the Automotive and Manufacturing Industry. In Proceedings of the 2019 30th Irish Signals and Systems Conference (ISSC), Maynooth, Ireland, 17–18 June 2019; pp. 1–6. [CrossRef]
28. Mitrea, D.; Tamas, L. Manufacturing Execution System Specific Data Analysis—Use Case With a Cobot. *IEEE Access* 2018, 6, 50245–50259. [CrossRef]
29. Krueger, V.; Chazoule, A.; Crosby, M.; Lasnier, A.; Pedersen, M.; Rovida, F.; Nalpantidis, L.; Petrick, R.; Toscano, C.; Veiga, G. A Vertical and Cyber–Physical Integration of Cognitive Robots in Manufacturing. *Proc. IEEE* 2016, 104, 1114–1127. [CrossRef]
30. Omron. Available online: http://www.omron-ap.com/robotics/mobilero/technology-page2.asp (accessed on 20 April 2020).
31. CTS-AIV. Available online: https://www.group-cts.de/en/robotics-inmatro/cts-AIV/ (accessed on 20 April 2020).
32. Ilas, C. Electronic sensing technologies for autonomous ground vehicles: A review. In Proceedings of the 2013 8th International Symposium on Advanced Topics in Electrical Engineering (ATEEE), Bucharest, Romania, 23–25 May 2013; pp. 1–6. [CrossRef]
33. Fetch Robotics—Freight (Research Unit). Available online: https://fetchrobotics.com/fetch-robotics-blog/research-edition-fetch-freight/ (accessed on 20 April 2020).
34. Clearpath—Ridgeback. Available online: https://clearpathrobotics.com/ridgeback-indoor-robot-platform/ (accessed on 20 April 2020).
35. Waypoint—Vector. Available online: https://waypointrobotics.com/vector-robotic-mobility-platform/ (accessed on 20 April 2020).
36. MIR-MIR100. Available online: https://www.mobile-industrial-robots.com/en/solutions/robots/mir100/ (accessed on 10 October 2020).
37. Pyo, Y.; Cho, H.; Jung, R.; Lim, T. ROS Robot Programming, 1st ed.; ROBOTIS Co.: Seoul, Korea, 2017; ISBN 979-11-962307-1-5.
38. Park, H.; Park, D. Real-time scheduler for Middleware of Industrial Robot. In Proceedings of the 14th Global Congress on Manufacturing and Management (GCMM-2018) Procedia Manufacturing, Brisbane, Australia, 5–7 December 2018; pp. 536–543.
39. Jeon, S.; Lee, J.; Kim, J. Multi-Robot Task Allocation for Real-Time Hospital Logistics. In Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC) Banff Center, Banff, AB, Canada, 5–8 October 2017; ISBN 978-1-5386-1645-1.
40. Song; Karray, F.; Li, H. A Real-time Scheduler Design for Fuzzy Logic Controller. In Proceedings of the IEEE International Conference on Mechatronics & Automation, Niagara Falls, ON, Canada, 20 July–1 August 2005.
41. Ham, A. Transfer-robot task scheduling in flexible job shop. J. Intell. Manuf. 2020, 31, 1783–1793. [CrossRef]
42. A ROS-Industrial Collaboration with Microsoft and BMW. Available online: https://rosindustrial.org/news/2018/11/28/a-ros-industrial-collaboration-with-microsoft-and-bmw. (accessed on 20 April 2020).
43. Node-RED. Available online: https://node-red.org/node/node-red-contrib-ros (accessed on 19 September 2020).
44. LabVIEW OPC UA Toolkit. Available online: https://www.ni.com/en-ie/support/downloads/software-products/download.labview-opc-ua-toolkit.html#305905 (accessed on 19 September 2020).
45. LabVIEW MQTT Toolkit. Available online: https://knowledge.ni.com/KnowledgeArticleDetails?id=ka00Z000000kKAcSAM&l=en-IE (accessed on 19 September 2020).
46. LabVIEW MQTT Toolkit. Available online: http://sine.ni.com/nips/cds/view/p/lang/en/nid/211065 (accessed on 19 September 2020).
47. Ioana, A.; Korodi, A. OPC UA Publish-Subscribe and VSOME/IP Notify-Subscribe Based Gateway Application in the Context of Car to Infrastructure Communication. Sensors 2020, 20, 4624. [CrossRef]
48. Eckhardt, A.; Müller, S.; Leurs, L. An evaluation of the applicability of OPC UA Publish Subscribe on factory automation use cases. In Proceedings of the IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA), Torino, Italy, 4–7 September 2018; pp. 1071–1074.
49. Kashyap, M.; Sharma, V.; Gupta, N. Taking MQTT and NodeMcu to IOT: Communication in Internet of Things. Procedia Comput. Sci. 2018, 132, 1611–1618. [CrossRef]
50. Caiza, G.; Llamuca, E.S.; Garcia, C.A.; Gallardo-Cardenas, F.; Lanas, D.; Garcia, M.V. Industrial Shop-Floor Integration Based on AMQP protocol in an IoT Environment. In Proceedings of the 2019 IEEE Fourth Ecuador Technical Chapters Meeting (ETCM), Guayaquil, Ecuador, 12–15 November 2019; pp. 1–6. [CrossRef]
51. Abukhalil, T.; Patil, M.; Sobh, T. A Comprehensive Survey on Decentralized Modular Swarm Robotic systems and Deployment Environments. Int. J. Eng. (IJE) 2013, 7, 44.
52. Węgierska, A.; Andrzejczak, K.; Kujawiński, M.; Granosik, G. Using LabVIEW and ROS for Planning and Coordination of Robot Missions, the Example of ERL Emergency Robots and University Rover Challenge Competitions. J. Autom. Mob. Robot. Intell. Syst. 2019, 13, 2. [CrossRef]
53. Omron Fleet-Manager. Available online: https://industrial.omron.eu/en/products/fleet-manager (accessed on 19 September 2020).

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).