IOP Conference Series:
Materials Science and
Engineering

PURPOSE-LED
PUBLISHING™

PAPER • OPEN ACCESS

Human-Robot Communication using Large Language Models for Automated Kitting

To cite this article: Mattias Bennulf *et al* 2026 *IOP Conf. Ser.: Mater. Sci. Eng.* **1342** 012049

View the article online for updates and enhancements.

You may also like

- Fluctuation-induced inward particle flux during L–I–H transition on HL-2A tokamak
J. Wu, T. Lan, W.X. Ding et al.
- Large language models integrated into brain–computer interfaces for communication and control: a systematic review
Alon Gorenshtein, Mahmud Omar, Yiftach Barash et al.
- MadEvolve: evolutionary optimization of cosmological algorithms with large language models
Tianyi Li, Shihui Zang and Moritz Münchmeyer

This content was downloaded from IP address 193.10.197.23 on 02/09/2026 at 14:01

IOP Conf. Series: Materials Science and Engineering **1342** (2026) 012049 doi:10.1088/1757-899X/1342/1/012049

Human-Robot Communication using Large Language Models for Automated Kitting

Mattias Bennulf [1a*], Fredrik Danielsson [1b] and Xiaoxiao Zhang [1c]

[1] Department of Engineering Science, University West, Trollhättan, Sweden

[a] https://orcid.org/0000-0002-7683-7662
[b] https://orcid.org/0000-0002-6604-6904
[c] https://orcid.org/0000-0003-1200-1951

[*] E-mail: mattias.bennulf@hv.se

Abstract. The requirements for rapidly changing product designs and product customisation increase the need for manufacturing systems to be updated frequently. Existing approaches, such as Plug & Produce, use standardised resources that can be moved around quickly when needed to adapt to these requirements. However, robot controllers still require reprogramming or extensive reconfiguration to work with the new setups. The use of a Large Language Model (LLM) has potential in assisting in automatically adapting these systems to new product designs. When changes are frequent, the use of in-house knowledge should be the focus rather than the use of external expert knowledge. Using LLM, a manufacturing system can be instructed on what to do using natural language. This simplifies and speeds up the changes needed to adapt to new product requirements. This article presents an implemented and tested system for using LLM with a physical collaborative robot equipped with a mechanical gripper and a vision system. This is tested with a kitting application that includes a set of buffers for holding the objects to be kitted, a human giving instructions to the system, and a kitting tray to hold the kit.

1. Introduction

Frequently changing product designs puts demand on the manufacturing systems that produce them. A shift can be identified in the manufacturing industry to focus more on mass customisation and low-volume manufacturing, rather than traditional mass production [1].

When working with mass customisation and low volumes, traditional automation systems are not suitable due to rigid hardware and control structures. Thus, there is a need for reduced installation costs [2], [3]. One approach is Plug & Produce, which was introduced by Arai et al. [4]. It makes it easy to add and remove hardware to the system by standardising modules [5], [6]. There have already been many efforts to solve the hardware challenges of Plug & Produce, as shown in, for example [7], [8]. Thus, the focus of this article is on the software-related challenges.

Flexible manufacturing with customised products or low-volume still relies on human involvement, both performing operations and instructing the system about changes. To reduce the time spent on re-programming and extensive re-configurations, humans should be able to instruct the system in a natural language. This will make the system faster to adapt to new requirements with a time frame of minutes to hours, rather than days to months. It also allows for

Content from this work may be used under the terms of the Creative Commons Attribution 4.0 licence. Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.
Published under licence by IOP Publishing Ltd

the use of in-house knowledge, where, for example, an operator or technician can instruct the system. Industry 5.0 is defined with a focus on a human-centric industry [9]. Lu et al. [10] explain that a human-centric approach puts the human in the centre rather than the system. To understand this, they define the industrial human needs divided into five levels of human-machine relationships: *(1) Coexistence, (2) Health*, *(3) Collaboration, (4) Compassion, (5) Coevolution.* Levels 1, 2, and 3 can be approached with existing technology for safety and collaboration. For the higher levels 4 and 5, there is a need for more intelligent approaches, such as Artificial Intelligence (AI).

Large Language Models (LLMs) represent a form of AI that enables a more natural interaction between humans and machines. One existing example where humans communicate with robots in natural language is to categorise the input (prompt) to the LLM into *system signals*, *system description* and *product description* [11]. The *system signals* contain information about the system, such as the robot gripper location and whether the gripper is open or closed. The *system description* is written by an expert who is involved when designing the manufacturing system. The *product description* should ideally be based on in-house knowledge and expressed in a natural language, which is particularly important when product specifications change frequently. LLMs can be used for generating texts, such as programming code [12] and to generate explanations of existing texts, such as programming code [13]. However, it is important to note that they do not use human-level logical and causal reasoning for generating the outputs, as explained by Manning [14].

To avoid manually defining the position of objects to be used in, for example, a pick-and-place scenario, depth-sensing cameras are a useful tool for robot manipulation. There are a variety of depth-sensing cameras to select from [15]. One approach presented by Le et al. [16] used the object detection system "You Only Look Once" (YOLO) to identify objects for a robot to pick and place. YOLO was first introduced in 2016 by Redmon et al. [17]. It is an object detection system based on neural networks and is one of the most popular frameworks for object detection. It can be trained on pictures from, for example, a manufacturing scenario to recognise objects. There are 12 versions of YOLO available between 2016 and 2025 [18]. Another object detection system is YOLO-World, which was introduced in 2024 by Cheng et al. [19]. It uses the YOLO architecture with a pre-trained "Contrastive Language-Image Pre-Training" (CLIP) text encoder for encoding the text inputs. CLIP is a neural network that has been trained using image and text pairs [20]. Thus, YOLO-World can be given a natural language input and detect those objects without the need for re-training. This is useful in a flexible manufacturing system to avoid the use of experts for re-training neural networks.

This article presents a novel system where a human without expert knowledge on manufacturing systems can instruct an industrial robot on what to do using natural language. The system is equipped with a depth-sensing camera that detects the location of objects. A prompt is automatically generated consisting of user inputs that are separated into in-house knowledge and expert knowledge, together with conversation history, machine vision data and states of the manufacturing system. This prompt is then automatically sent to the LLM that is controlling the robot. This is tested in a physical experiment in a kitting application using an industrial robot equipped with a mechanical gripper that picks metal screws from buffers and places them in a kitting tray.

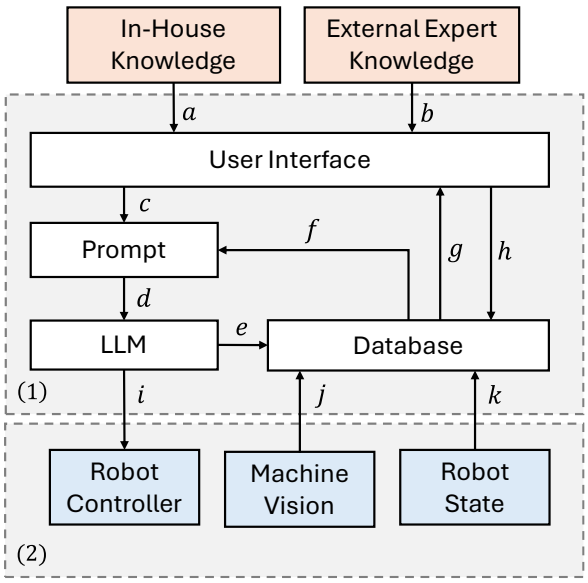

Figure. 1. Proposed LLM Decision-Making System (1) and Plug & Produce manufacturing system (2). The $a, b, \ldots, k$, defines the information flow between the nine containers

2. Proposed System

In Figure. 1, the proposed LLM Decision-Making System (1), and the Plug & Produce manufacturing system (2) are shown with arrows a,b,…,k, that define the information flow between the following nine containers:

In-House Knowledge is a human within the company, with manufacturing knowledge, who instructs the system *(a)* in a natural language.

External Expert Knowledge refers to a specialised individual, typically external to the company, who possesses a deep expertise in developing the manufacturing system and with strong understanding of robotics and automation. This user gives instructions *(b)* in natural language about how the system works. This can, for example, include defining locations of objects, defining how to perform a pick and place sequence and whether it needs to go via offset locations.

User Interface is a graphical interface that runs on, for example, a web service to allow it to be compatible with most devices and to run from a distance, such as on a tablet computer in the manufacturing system. It sends information *(c)* to *Prompt* and stores the user input *(h)* in the *Database*. The *User Interface* also reads *(g)* and presents information from the *Database* to the user.

Prompt is a software that combines the user interface input *(c)* with information from the database *(f)* and generates a prompt *(d)* for the LLM.

LLM generates a decision on controlling the robot and sends instructions *(i)* to the *Robot Controller* and stores history *(e)* in the *Database.*

Database receives information *(j)* from the *Machine Vision* system and *(k)* from the *Robot State*.

Robot Controller is a physical or simulated robot that takes the LLM instructions *(i)* and runs them. This includes, for example, moving the robot Tool Center Point (TCP) and opening and closing the gripper of the robot.

Machine Vision system consists of a camera and a function to convert an image to a set of objects described in text with required details such as location and type.

Robot State is a software that captures the state of the robot and converts it to text. This can, for example, include the current location of the TCP of the robot and the gripper's open and closed states.

In Figure. 2, a simplified illustration of the *User Interface* is shown. It includes a conversation function with a *(1) Conversation History*, *(2) Message Input,* and *(3) Send Button*. It also shows the *(4) Robot State* with *x, y, z* coordinates, the gripper open/close state, and a description in natural language of where the robot tool is located. There is also an *(5) Event Log* that shows all actions that the LLM has sent to the *Robot Controller.*

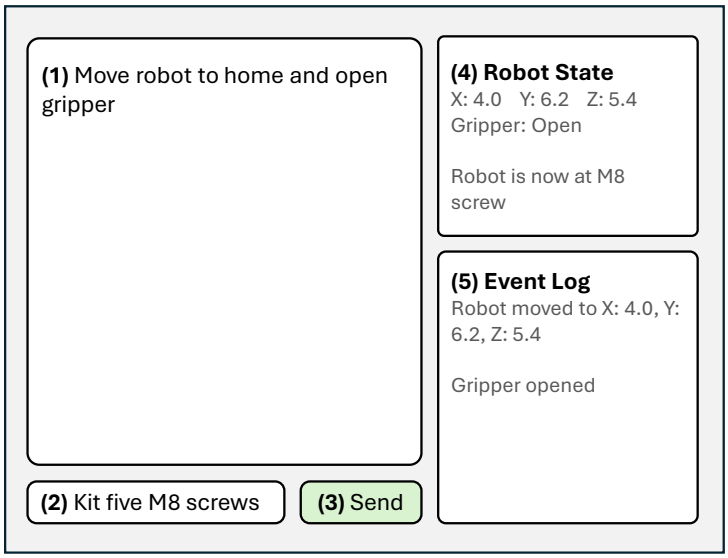

Figure. 2. User Interface with (1) Conversation History, (2) Message Input, (3) Send Button, (4) Robot State, and (5) Event Log

The machine vision system must be able to identify all objects to be manipulated by the robot. This includes the object location, orientation and object type. The vision system should not require frequent re-training. This is especially important in a Plug & Produce system where the objects are changed frequently. The *Machine Vision* system must output information in a text format that can be understood by an LLM.

3. Implementation

The proposed system is implemented as shown in Figure. 3, containing an industrial robot (Franka Robotics), a 3D camera (Intel RealSense), a kitting tray, and parts to be kitted. The parts are two types of screws with the sizes 8 and 14 millimetres in diameter (M8, M14).

The robot has a path planner running in the software *MoveIt Motion Planning Framework* (MoveIt), which is a tool in the *Robot Operating System* (ROS). For an industrial setup, all geometries should be added in detail to the simulation to avoid collisions. For the experiments, the simulation model was simple and did not include all obstacles.

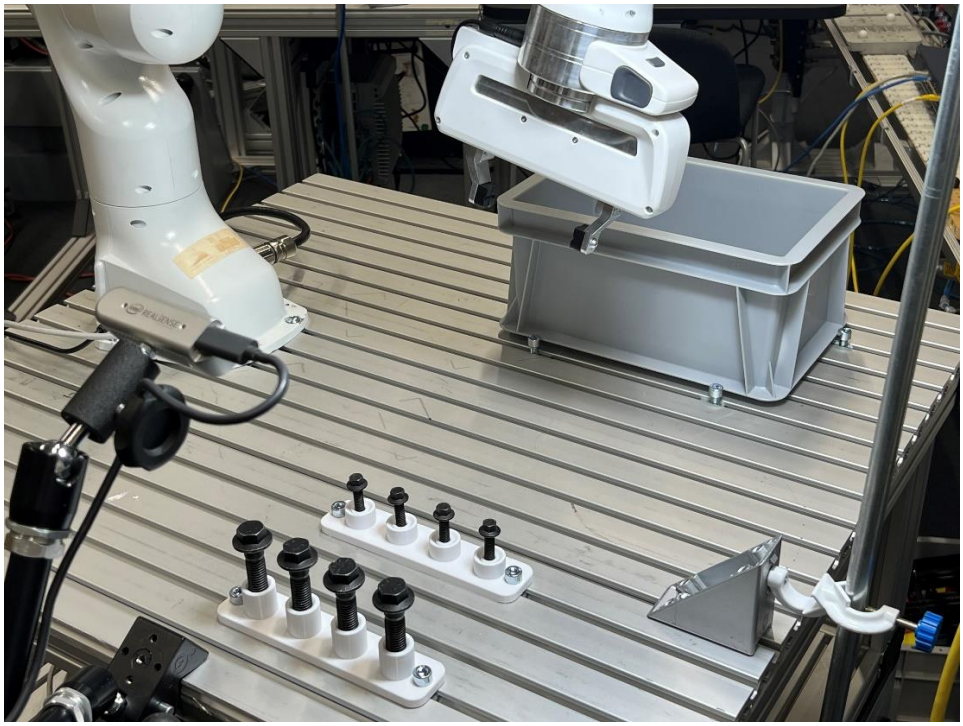

Figure. 3. Industrial robot (Franka Robotics), 3D-camera (Intel RealSense), kitting tray and parts to be kitted

Since LLM models take text as input, the photos taken with the camera must be converted into text. We considered using the object detection system *You Only Look Once* (YOLO), and later selected YOLO-World. Both systems can provide the positions and heights of objects required for robot manipulation. The key difference is that YOLO needs to be trained for the detection of scenario-specific objects. This requires manually marking the location of items on hundreds to thousands of images. Using YOLO-World, this can be completely avoided since it is pre-trained on a general dataset. However, YOLO outputs a list of objects based on the camera input. This is not the case in YOLO-World, where the system requires a keyword to be entered that it searches for in the picture. The syntax for these keywords can be difficult to know. For the experiments in this article, YOLO-World was used. This means that we did not have to train the system, but we had to manually find the exact prompts. In the future, this can be investigated further to let the LLM match user input, like pick M14 screw to the specific YOLO-WORLD prompt required.

The user interface was developed as a web page, as shown in Figure. 4, and Figure. 5. The web page connects to a database running on the same server as the web page. This makes it easy to connect to it from any device in a factory, such as a tablet computer, mounted next to the robot. The message that the user sends is not just a simple instruction. Instead, it is a conversation in chat format that has the possibility to be used for general topics, making it possible to ask questions about the system.

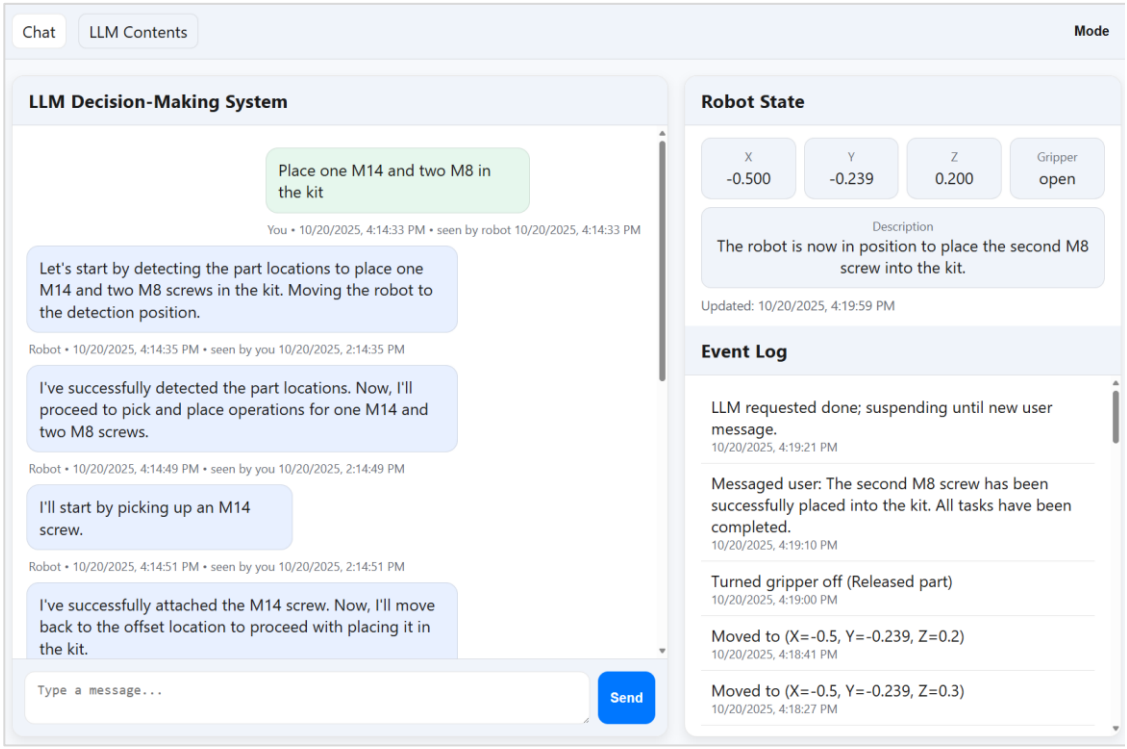

Figure. 4. User Interface for the LLM Decision-Making System, showing data from the experiment

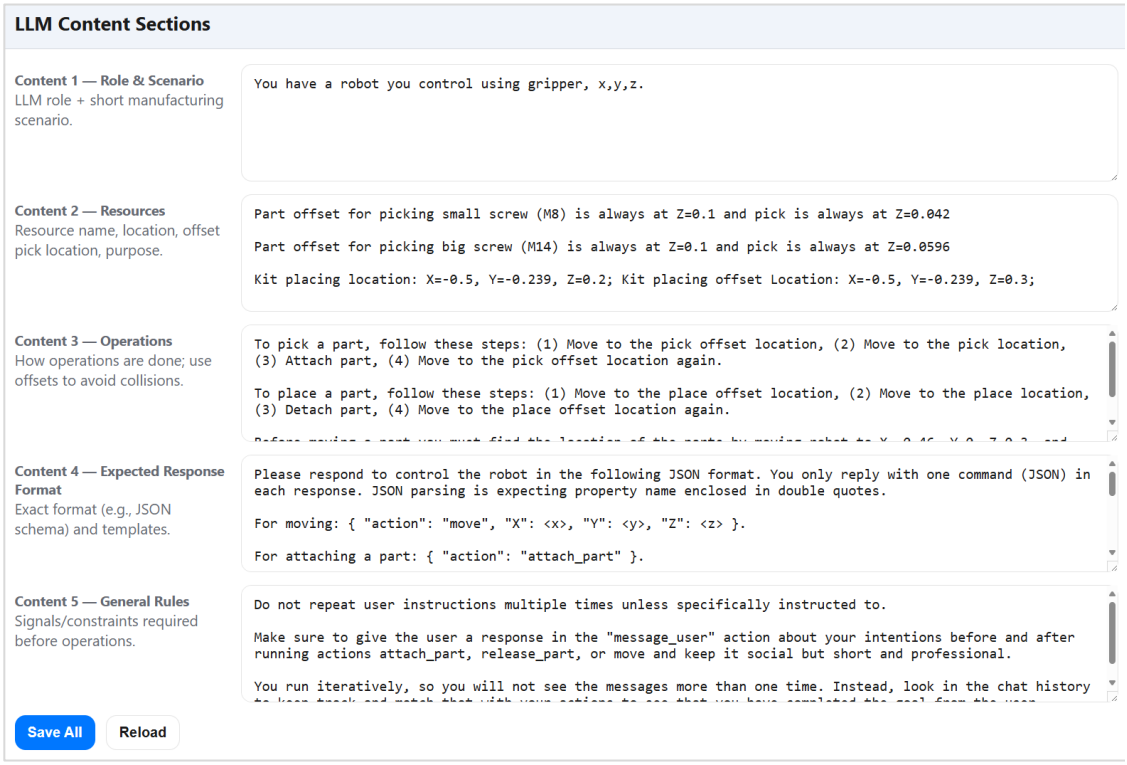

Figure. 5. The LLM Contents Editor with five contents and their descriptions, and the content data used in the experiment

The selected LLM model was gpt-4o, which was used for the experiment. The prompt was divided into a system and user role due to the model's design to separate these. The rest of this section shows the prompts that are considered to be expert knowledge. The italic, non-bold text is the prompt given to the LLM, and when brackets *{variable}* are used, the brackets are replaced by the actual value of that variable.

System role:
Your role is:
{Content 1}
The system has the following resources:
{Content 2} {Camera}
You can perform the following operations:
{Content 3}
Expected response format:
{Content 4}
General rules for system:
{Content 5}

User role:
{Signals}
{User Input}
{Chat History}

Content 1:
You have a robot you control using gripper, x,y,z.

Content 2:
Part offset for picking small screw (M8) is always at Z=0.1 and pick is always at Z=0.042
Part offset for picking big screw (M14) is always at Z=0.1 and pick is always at Z=0.0596
Kit placing location: X=-0.5, Y=-0.239, Z=0.2; Kit placing offset Location: X=-0.5, Y=-0.239, Z=0.3;

Content 3:
To pick a part, follow these steps: (1) Move to the pick offset location, (2) Move to the pick location, (3) Attach part, (4) Move to the pick offset location again.

To place a part, follow these steps: (1) Move to the place offset location, (2) Move to the place location, (3) Detach part, (4) Move to the place offset location again.

Before moving a part you must find the location of the parts by moving robot to X=-0.46, Y=0, Z=0.3, and then send the "detect" action to update the location data of the parts.

Content 4:
Please respond to control the robot in the following JSON format. You only reply with one command (JSON) in each response. JSON parsing is expecting property name enclosed in double quotes.

For moving: { "action": "move", "X": <x>, "Y": <y>, "Z": <z> }.

For attaching a part: { "action": "attach_part" }.

For releasing a part: { "action": "release_part" }.

For replying to "user input" { "action": "message_user", "message": <message> }.

For waiting on a resource to finish { "action": "wait" }.

For detecting part run { "action": "detect" }.

Run when you are not having a plan to perform any further actions right now { "action": "done" }.

After each action move, send description on where the robot is located without using x,y,z { "action": "Location", "message": <message> }.

Content 5:
Do not repeat user instructions multiple times unless specifically instructed to.

Make sure to give the user a response in the "message_user" action about your intentions before and after running actions attach_part, release_part, or move and keep it social but short and professional.

You run iteratively, so you will not see the messages more than one time. Instead, look in the chat history to keep track and match that with your actions to see that you have completed the goal from the user.

Signals:
Current crane position, X={atX}, Y={atY}, Z={atZ}
Current gripper status, gripper={gripper_status}

***User Input**:*
The following is the "user input" giving you input on what to do in the system. If this is empty send the "done" command and do not do anything:
{UserInputFromDb}

***Chat History**:*
Chat history:{History}

Camera (generated):
The LLM Decision-Making System gets data from the vision system, which consists of a camera, YOLO-World and a REST server used for communication. The LLM Decision-Making System loops through the data, which consists of x and y locations for each screw detected. The data is formatted with name and location (Name: <name>, x=<x>, y=<y>) for each screw and appended to the string variable *Camera*.

Table. 1. The observed behaviours of the robot and the correlated actions

Action	Behaviour
move	Move to avoid covering the camera
move	Move to the offset of an M14 screw
move	Move to the M14 screw
attach_part	Attach the M14 screw
move	Move to the offset of the M14 screw
move	Move to the offset of the kit
move	Move to kit
release_part	Release the part in the kit
move	Move to the offset of an M8 screw
move	Move to the M8 screw
attach_part	Attach the M8 screw
move	Move to the offset of the M8 screw
move	Move to the offset of the kit
move	Move to kit
release_part	Release the part in the kit
move	Move to the offset of an M8 screw
move	Move to the M8 screw
attach_part	Attach the M8 screw
move	Move to the offset of the M8 screw
move	Move to the offset of the kit
move	Move to kit
release_part	Release the part in the kit

4. Results

In Table. 1, the behaviour of the robot is listed. The list was created manually by observing the robot's behaviour and is limited to the physical actions: *attach_part*, *release_part*, and *move*. This was done to evaluate the actual behaviour of the system rather than the event log of the system. Thus, the actions *detect*, *location*, *wait*, *message_user*, and *done* are not included in this list.

It was noticed that the prompts need to be tested and adjusted to improve the robot control. At first, there was a problem with the offsets not being included correctly when picking screws. This was solved by clarifying the prompt as shown in the article, using clear language like: *pick offset location* and *place offset location.* Earlier, this was only *offset location.* However, in the final tests, it was noticed that the system did not use the offset when leaving the kit. This can likely be solved in the same way by modifying the prompt or by adding the kit to the path planner, in this case, MoveIt. In fact, many offset locations can be avoided using the path planner, making the prompt shorter and easier for the LLM to understand.

5. Conclusion

This article shows a system that is easy to adapt to new product designs using in-house knowledge. The implementations show how a user interface can be designed with a chat format, letting the user communicate with an AI model. The results show that the system can take

operator instructions in a natural format and combine them with the expert user's input to control the robot.

When the system is to be installed, there is a requirement for expert knowledge. This was clear when designing the prompt with expert knowledge. To make the prompts to describe the system, there is a need to test and edit the prompts, similar to the approach of most software development. All resources that should have communication with the LLM need to be added with functions corresponding to the prompt. In the future, the LLM Decision-Making System could be more general to automatically connect several device types and protocols to completely avoid programming at system installation.

Using a vision system that requires no model training saves time for new product types. However, the accuracy observed in YOLO-World was less than in YOLO. Further, the system still required an expert to adapt the prompts for the vision system. In the future, an investigation of other vision systems and approaches should be conducted.

In the experiments, the LLM performed some planning. In the future, it is of interest to test more advanced scenarios that require complex planning, such as placing two items overlapping and asking the system to remove the one below and letting the system figure out that it must remove the one on top first to complete this. This can be tested with an LLM or with a traditional planner connected to the LLM to avoid planning errors. This would let the LLM mainly focus on interpreting the user input rather than planning.

6. Acknowledgements

The work presented in this article was funded by the KK-foundation synergy project PROWIL Dnr. 20230032. This work was also funded by Swedish Governmental Agency for Innovation Systems (Vinnova) under the project named AIKIDO (Dnr: 2025-01009). Their support is gratefully acknowledged.

References

[1] S. J. Hu *et al.*, 'Assembly system design and operations for product variety', *CIRP Annals*, vol. 60, no. 2, pp. 715–733, 2011, doi: 10.1016/j.cirp.2011.05.004.

[2] H. A. ElMaraghy, 'Flexible and reconfigurable manufacturing systems paradigms', *International Journal of Flexible Manufacturing Systems*, vol. 17, no. 4, pp. 261–276, Oct. 2005, doi: 10.1007/s10696-006-9028-7.

[3] T. Blecker and N. Abdelkafi, 'Mass Customization: State-of-the-Art and Challenges', in *Mass Customization: Challenges and Solutions*, vol. 87, Boston: Springer, Boston, MA, 2006, pp. 1–25. doi: 10.1007/0-387-32224-8_1.

[4] T. Arai, Y. Aiyama, Y. Maeda, M. Sugi, and J. Ota, 'Agile Assembly System by "Plug and Produce"', *CIRP Annals*, vol. 49, no. 1, pp. 1–4, 2000, doi: 10.1016/S0007-8506(07)62883-2.

[5] M. Onori, N. Lohse, J. Barata, and C. Hanisch, 'The IDEAS project: plug & produce at shop-floor level', *Assembly Automation*, vol. 32, no. 2, pp. 124–134, Apr. 2012, doi: 10.1108/01445151211212280.

[6] C. Schou and O. Madsen, 'A plug and produce framework for industrial collaborative robots', *Int J Adv Robot Syst*, vol. 14, no. 4, Jul. 2017, doi: 10.1177/1729881417717472.

[7] A. Zoitl, G. Kainz, and N. Keddis, 'Production Plan-Driven Flexible Assembly Automation Architecture', in *Industrial Applications of Holonic and Multi-Agent Systems*, 2013, pp. 49–58. doi: 10.1007/978-3-642-40090-2_5.

[8] L. Ribeiro, J. Barata, M. Onori, and J. Hoos, 'Industrial Agents for the Fast Deployment of Evolvable Assembly Systems', in *Industrial Agents*, Elsevier, 2015, pp. 301–322. doi: 10.1016/B978-0-12-800341-1.00017-6.

[9] European Commission, Directorate-General for Research and Innovation, M. Breque, L. De Nul, and A. Petridis, 'Industry 5.0 – Towards a sustainable, human-centric and resilient European industry', *Publications Office of the European Union*, 2021, [Online]. Available: https://data.europa.eu/doi/10.2777/308407

[10] Y. Lu *et al.*, 'Outlook on human-centric manufacturing towards Industry 5.0', *J Manuf Syst*, vol. 62, pp. 612–627, Jan. 2022, doi: 10.1016/j.jmsy.2022.02.001.

[11] M. Bennulf and F. Danielsson, 'Using Large-Language Models for Plug & Produce Manufacturing', in *23rd International Industrial Simulation Conference, ISC 2025*, Skovde: EUROSIS, ISBN: 978-949285935-8, Jun. 2025, pp. 43–47.

[12] M. Nejjar, L. Zacharias, F. Stiehle, and I. Weber, 'LLMs for science: Usage for code generation and data analysis', *Journal of Software: Evolution and Process*, vol. 37, no. 1, Jan. 2025, doi: 10.1002/smr.2723.

[13] D. Nam, A. Macvean, V. Hellendoorn, B. Vasilescu, and B. Myers, 'Using an LLM to Help With Code Understanding', in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, New York, NY, USA: ACM, Apr. 2024, pp. 1–13. doi: 10.1145/3597503.3639187.

[14] C. D. Manning, 'Human Language Understanding & Reasoning', *Daedalus*, vol. 151, no. 2, pp. 127–138, May 2022, doi: 10.1162/daed_a_01905.

[15] L. Rustler, V. Volprecht, and M. Hoffmann, 'Empirical Comparison of Four Stereoscopic Depth Sensing Cameras for Robotics Applications', *IEEE Access*, vol. 13, pp. 67564–67577, 2025, doi: 10.1109/ACCESS.2025.3560810.

[16] H. Le, X. Phan, T. Phan, and T. Tran, 'Development of an intelligence vision for a robot system to pick and place objects', *FME Transactions*, vol. 53, no. 2, pp. 233–242, 2025, doi: 10.5937/fme2502233P.

[17] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, 'You Only Look Once: Unified, Real-Time Object Detection', in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2016, pp. 779–788. doi: 10.1109/CVPR.2016.91.

[18] J. Wei, A. As'arry, K. Anas Md Rezali, M. Zuhri Mohamed Yusoff, H. Ma, and K. Zhang, 'A Review of YOLO Algorithm and Its Applications in Autonomous Driving Object Detection', *IEEE Access*, vol. 13, pp. 93688–93711, 2025, doi: 10.1109/ACCESS.2025.3573376.

[19] T. Cheng, L. Song, Y. Ge, W. Liu, X. Wang, and Y. Shan, 'YOLO-World: Real-Time Open-Vocabulary Object Detection', in *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2024, pp. 16901–16911. doi: 10.1109/CVPR52733.2024.01599.

[20] A. Radford *et al.*, 'Learning Transferable Visual Models From Natural Language Supervision', Feb. 2021, doi: https://doi.org/10.48550/arXiv.2103.00020.