DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg
Campus Friedrichshafen

SCHRIFTENREIHE DER FAKULTÄT FÜR TECHNIK
DER DUALEN HOCHSCHULE BADEN-WÜRTTEMBERG
RAVENSBURG

2021/01

Robotics – An Educational Perspective

Harsh Sheth M.Sc., Kris Dalm MBA, Rohan Sahuji M.Sc.,
Prof. Dr.-Ing. Thomas Dietmüller, Prof. Dr.-Ing. Lars Ruhbach,
B.Eng. Niklas Hohenauer, B.Eng. Matthias Münsch

DHBW
Duale Hochschule
Baden-Württemberg
Ravensburg
Campus Friedrichshafen

SCHRIFTENREIHE DER FAKULTÄT FÜR TECHNIK DER DUALEN HOCHSCHULE BADEN-WÜRTTEMBERG RAVENSBURG

2021/01

Robotics – An Educational Perspective

Harsh Sheth M.Sc., Kris Dalm MBA, Rohan Sahuji M.Sc., Prof. Dr.-Ing. Thomas Dietmüller, Prof. Dr.-Ing. Lars Ruhbach, B.Eng. Niklas Hohenauer, B.Eng. Matthias Münsch

IMPRESSUM

Schriftenreihe der Fakultät für Technik
der Dualen Hochschule Baden-Württemberg Ravensburg

Herausgeber

Prof. Dr. Heinz-Leo Dudek
Prorektor und Dekan der Fakultät für Technik

Duale Hochschule Baden-Württemberg Ravensburg
Baden-Wuerttemberg Cooperative State University
Marienplatz 2
88212 Ravensburg
Deutschland

http://www.ravensburg.dhbw.de

2021/01, Januar 2021

ISBN 978-3-945557-01-3
ISSN 2199-238X
DOI 10.12903/DHBW_RV_FN_01_2021_SHETH_DALM_SAHUJI_DIETMUELLER_RUHBACH_HOHENAUER_MUENSCH

Druck und Verarbeitung

Gestaltung
Nicole Stuepp
DHBW Ravensburg
Marienplatz 2, 88212 Ravensburg

Druck
Frick Kreativbüro & Onlinedruckerei e.K.
Brühlstraße 6
86381 Krumbach

Common Abbreviation

DLL	Dynamic Link Library
TCP	Tool Center Point
I/O	Input/Output
DHCP	Dynamic Host Configuration Protocol
DOF	Degree of Freedom
UI	User Interface
GUI	Graphical User Interface
API	Application Programming Interface

Robotics – An Educational Perspective

Autors	**Title**
Harsh Sheth M.Sc.[1] Kris Dalm MBA[2] Rohan Sahuji M.Sc.[3]	Efficient Operation and Visualization of Robots.
Prof. Dr.-Ing. Thomas Dietmüller[4] Prof. Dr.-Ing. Lars Ruhbach[5] B.Eng. Niklas Hohenauer[6] B.Eng. Matthias Münsch[7]	A Setup using Cobots for Training both CPS and Sensorics in Engineering Education.

[1] IWT Wirtschaft und Technik GmbH, Project Manager
[2] IWT Wirtschaft und Technik GmbH, Area Manager; Dozent der DHBW Ravensburg
[3] IWT Wirtschaft und Technik GmbH, Project Engineer
[4] DHBW Ravensburg Campus Friedrichshafen, Professor Maschinenbau
[5] DHBW Ravensburg Campus Friedrichshafen, Studiengangsleiter Maschinenbau; CEO IWT Wirtschaft und Technik GmbH
[6] Absolvent der DHBW Ravensburg im Studiengang Maschinenbau
[7] Absolvent der DHBW Ravensburg im Studiengang Maschinenbau

Efficient Operation and Visualization of Robots.

Harsh Sheth M.Sc.[8], Kris Dalm MBA[9], Rohan Sahuji M.Sc.[10]

Keywords:

Robots, Collaborative Robots, Graphical User Interface, Trajectory planning

[8] IWT Wirtschaft und Technik GmbH, Project Manager

[9] IWT Wirtschaft und Technik GmbH, Area Manager; Dozent der DHBW Ravensburg

[10] IWT Wirtschaft und Technik GmbH, Project Engineer

1 INTRODUCTION AND AIM

Today, Industrial Robots are on the verge of revolutionizing manufacturing. As they become smarter, faster, and cheaper, they are being called upon to do more. They are taking on more “human” capabilities and traits such as sensing, dexterity, memory, and trainability. As a result, they are taking on more tasks - such as picking and packaging, testing, or inspecting products, or assembling minute electronics. The new World Robotics report by the International Federation of Robotics (IFR) shows an annual global sales value of 16.5 billion USD in 2018 – a new record. 422,000 units were shipped globally in 2018 - an increase of 6 percent compared to the previous year. They also predict an average growth of 12 percent per year from 2020 to 2022. Europe´s robot installations rose by 14 percent, with Germany being the largest in Europe and the fifth largest in the world. In 2018, the number of robots sold in Germany increased by 26% to almost 27,000 units. [1]

One of the main problems in industrial and collaborative robots is the fact that these robots are difficult to program and integrate in an industrial application. Many of the robots have prerequisites of knowing certain programming languages in order to operate the robot. While this method provides a greater flexibility to the robot operator, it is only advantageous if they know the required programming language.

The aim of this Project was to develop such an interface for the current collaborative robots on the market, which allows the user to use the robot without any prior knowledge of programming. Along with this, a Dynamic Link Library (DLL) file was developed which contained the necessary functions for robot operations, for students and people having programming experience who want to have a more technical grasp on the robot or want to program the robot remotely.

This Project was carried out as a Master Thesis under Project Lernfabrik Fallenbrunnen in IWT Wirtschaft und Technik GmbH. Due to the time constraint on the Project, it was carried out on the Universal Robot UR5 Collaborative Robot with a OnRobot RG2 two finger Gripper and with Visual Studio [2] Integrated Development Environment (IDE) with the programming language C#.

Although the Universal Robot has a teaching pendant to program it, it makes it difficult when a need for offline or remote programming of the robot or a requirement to program it externally to connect with other devices or robot arises. This interface was developed with UR5 robot but can be used with any robot of the UR series, as they have the same programming and interfacing possibility.

2 HARDWARE USED

Universal Robot UR5

Universal Robots UR5 [3] is the second smallest version of the four types of robots provided by Universal Robots. It has six joints, that is six degrees of freedom and has a reach of 850mm. This robot can operate at a maximum velocity of 5000 mm/s at Tool Centre Point (TCP) in certain scenarios, but those scenarios are dangerous for collaborative operations. It has the repeatability of ±0.1 mm. This robot has 16 Digital I/Os and 2 Analog I/Os. It has a payload of 5kg and has safety functions certified by TüV NORD. This robot comes with a teaching pendant which can be utilized for easy and quick programming of the robot. It has a free drive mode which can be used for hand guiding the robotic arm for waypoint definition.

This robot can operate at a typical tool speed of 1000 mm/s but maximum can be around 4000 mm/s to 5000 mm/s depending on the operation. Robot operation at such speeds is dangerous as there are no external sensors or cameras to detect a human or obstacle presence. The robot cannot be stopped with only lightly touching the robot, but a considerable amount of force needs to be given to stop it. According to ISO 10218-1:2011 [4], the maximum speed when a human is in the working area of the robot is 250 mm/s.

There are mainly two different approaches or ways to program a Universal-Robot – either with the Polyscope GUI interface using the touch screen teach pendant – or using the UR script language – or a combination of both the approaches. Script programming on the UR can be performed by sending raw script commands from an external device to the robot – even without any program made in Polyscope on the robot.

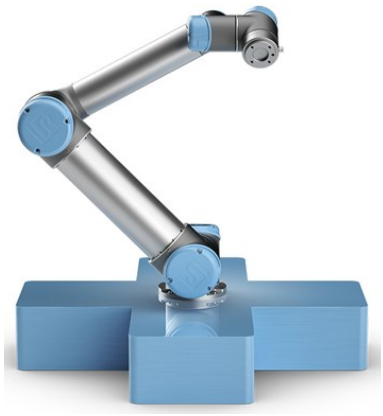

Figure 1: Universal Robot UR5[11]

[11] https://www.universal-robots.com/products/ur5-robot/

OnRobot RG2 Gripper

The gripper used in this is the RG2 Gripper from OnRobot [5]. It has a payload of 2kg with a gripping force of 3-40N. The maximum width that this gripper can open is up to 110mm but with the offset from the fingers, it is around 105mm. This gripper comes with a software which is installed in the teaching pendant providing easy installing and easy use of this gripper using the teaching pendant.

Figure 2: OnRobot RG2 Gripper[12]

Type of Motions in UR Robot

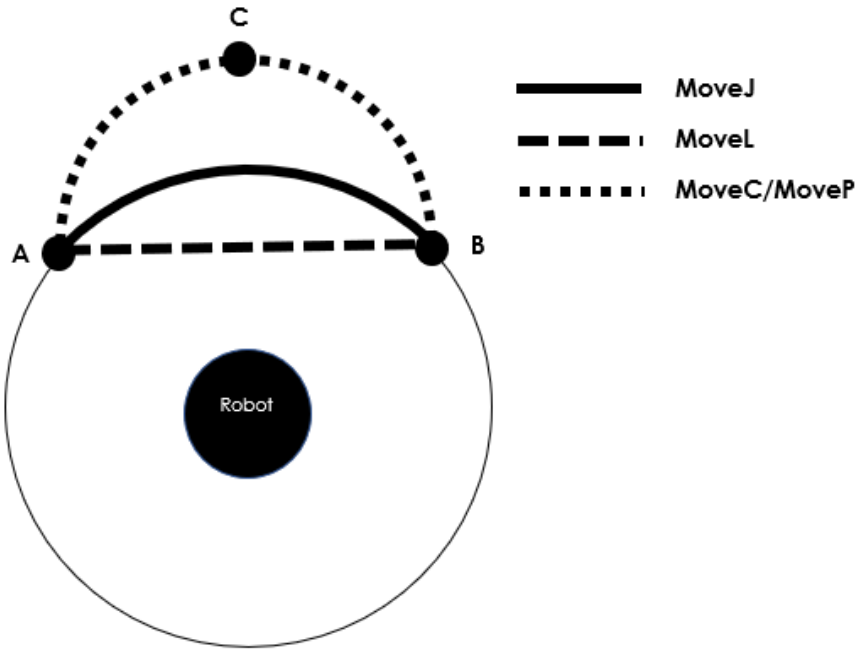

Figure 3: Different move commands in UR Robot

[12] https://onrobot.com/en/products/rg2-gripper, accessed on 21.04.2017

1) MoveJ

Perform a robot movement which is linear in joint space. As shown in Figure 3, the path in solid black is the path followed by the robot, if a moveJ command is given. This type of movement is optimal as least amount of joint movement is done.

2) MoveL

Perform a TCP linear move. As shown in Figure 3, the path shown in dashed line is the path followed by the robot, if a moveL command is given.

3) MoveC / MoveP

Perform a circular move. As shown in Figure 3, the dotted curve is the path followed by the robot, if a moveC command is given. In this command, a via_pose is given, in this figure as point C.

3 DEVELOPMENT OF USER INTERFACE (UI)

Initial Design of User Interface (UI)

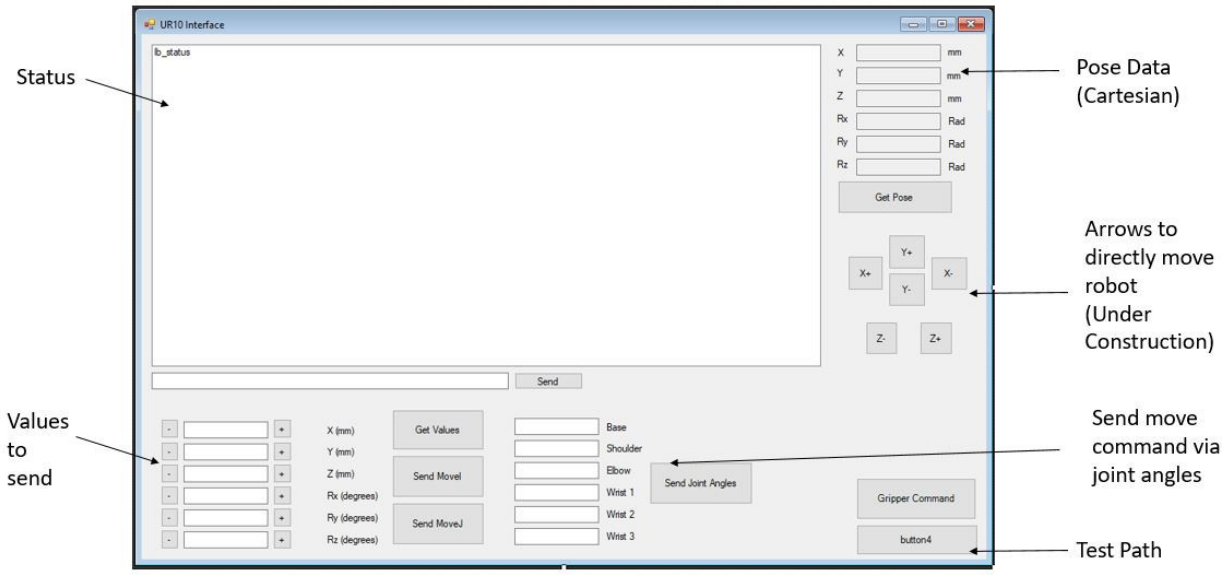

Figure 4: Initial User Interface Design

The design in Figure 4 is the initial design of the User Interface that was developed for this robot. As it is seen, the UI was in its primitive stages. It has a textbox which shows the status messages of the UI and responses from the robot's side. The current position of the robot could be viewed as well as custom values could be given to the robot to move it to a certain position in a 3-Dimensional Coordinate Space.

Final Design of User Interface (UI)

The design below shown in the Figure 5 is the final design of the UI for the UR Robot and is connected to the robot in real time over WiFi or Ethernet cable with TCP/IP Protocol. For connection via WLAN, IPv4 settings must be changed to Static instead of Dynamic Host Configuration Protocol (DHCP). The same Default Gateway and Subnet must be entered in the robot settings and the last part of the IP address has to be different than that of the robot, for example “192.168.0.xx”. The port to connect to the robot is either 30001,30002 or 30003.

For the robot, under “Setup Robot→Setup Network”, the network method should be Static, and Subnet and Gateway should be same as in IPv4 settings of the computer where the User Interface will be used.

Basic movement commands can be given and position, and joint data can be obtained from the robot. Path definition along with gripper operations are also possible through this interface. This interface monitors the position of the robot as well as the values of each robot joint and gives a warning when it is reaching its limit. Individual axis movements according to the distance provided can also be performed.

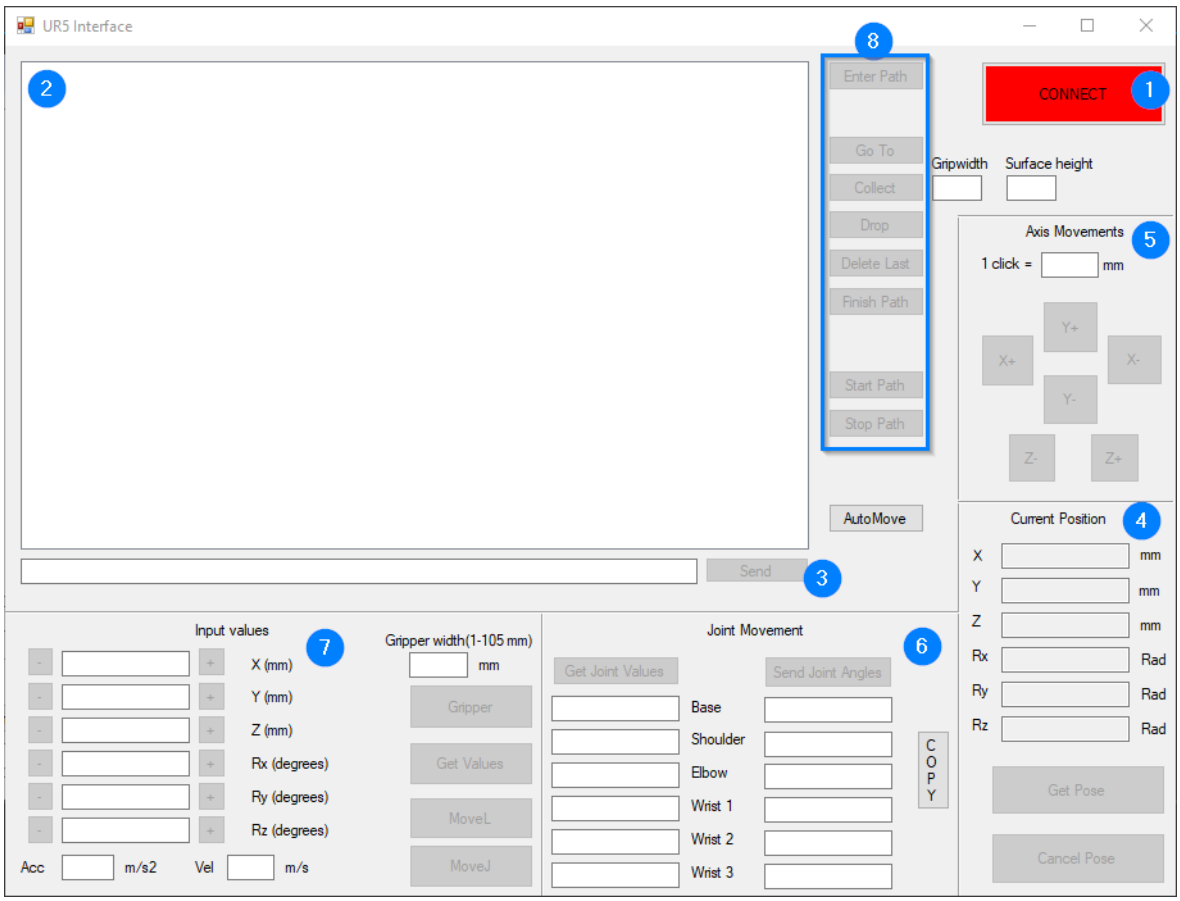

Figure 5: Final Design of the User Interface

Functions of the User Interface

This section explains all the functions in the Interface, shown in Figure 5.

1. **Connect:** Connect with UR5 robot over Wi-Fi.
2. **Status Box:** Listbox, which shows the status of the robot and coordinates where the robot is travelling.
3. **Command Box:** With this, you can send single script codes directly to the robot. E.g. movej(p[0.25,0.25,0.1,3.14,0,0],0.7,0.7,0,0) results in an arc move to the coordinate (250,250,100) with velocity 0.7m/s and acceleration 0.7m/s^2.
4. **Current Position:** This section shows the current absolute position of the robot.
    - **Get Pose:** Activates this feature.
    - **Cancel Pose:** Cancels this feature.
5. **Axis Movements:** These buttons will allow a single relative motion on the written axis. The distance to move must be provided in the textbox field.
6. **Joint Movements:** This robot is a 6 Degrees of Freedom (DOF) Robot, meaning it has 6 joints. This section will allow you to enter values for each joint (in degrees) to make a movement from joint values.
    - **Get Joint Values:** will get the current angles of all the joints with a safety warning when angles of any joint go above 345 degrees or below -345 degrees.
    - **Copy:** This button will copy the values from robot joint angles section to the input values section.
    - **Send Joint Angles:** This button sends the joint angles as a method to move.

    (Movement by this method is not advised. This section is just for information.)
7. **Input Values:** With this section, you can send the robot to the desired co- ordinate in 3D space along with desired acceleration and velocity.
    - **Get Values:** This button will copy the latest values from current position section to the textboxes.
    - **MoveL:** This button will initiate a linear move.
    - **MoveJ:** This button will result in an arc movement.
    - **Gripper:** This button will open or close the gripper to the width provided in the text field.

    Note: For velocity, it is advised to operate the robot maximum at 0.250m/s for collaborative operation.
8. **Robot Path Formation and Selection:** With this you can enter and save a new path for the robot to run.
    - **Enter Path:** Enter a new path to define. (Note: Always save the file with extension .txt)

- **Go To:** Enter coordinates in input values system with acceleration and velocity and click this button for defining a waypoint.
- **Delete Last:** Delete last command given in the path.
- **Finish Path:** Exit path editing mode.
- **Start Path:** Choose the desired .txt path file to run and start the movement of the robot.
- **Stop Path:** Stop the running path.
- **Drop:** Enter coordinates in input values system along with Gripwidth and Surface Height and click this button in order to command the robot to go to certain position and drop an object.
- **Collect:** Enter coordinates in input values system along with Grip- width and Surface Height and click this button in order to command the robot to go to certain position and collect an object.
    - **Gripwidth:** Width of the object to be picked up by the gripper. (Maximum 105mm)
    - **Surface Height:** Height of the surface for the robot to go to collect the object.

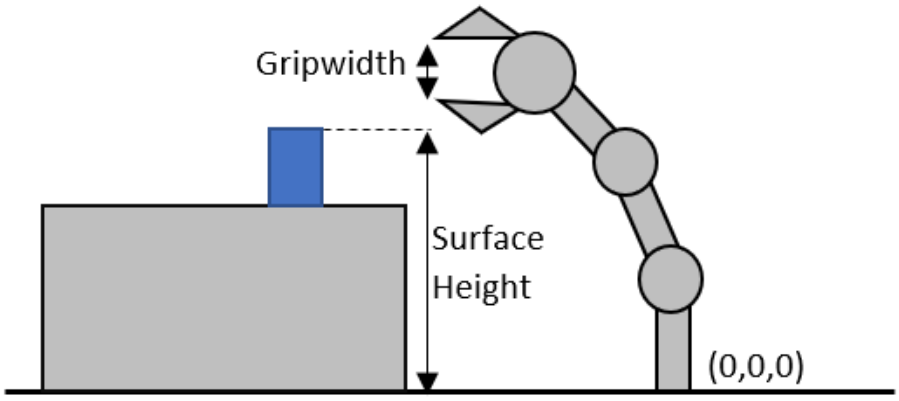

Figure 6: Scenario for Gripwidth and Surface height explanation[13]

Pose Data Acquisition

The pose of the robot is obtained in six values: *X, Y, Z, Rx, Ry, Rz*. *X, Y,* and *Z* are the position of the robot in 3-Dimensional space and *Rx, Ry* and *Rz* are the rotational positions of the robot. Normally, data acquisition is a simple task where there exists a command which asks the robot for the updated position. But for UR Robot, it sends a package of 1044 bytes whenever a socket is opened. In this 1044 bytes of data contains all possible information of the robot.

[13] Own illustration

All information obtained from the robot is found in a document called "client_interface.xlsx" that is found on the UR Robot website. In this list, the Tool Vector Actual part contains the position of the robot. There are 6 packets, each of 8 bytes containing the values for *X, Y, Z, Rx, Ry,* and *Rz*. The values are extracted but then it needs to undergo many conversions. The bytes are first converted to String, then from String to Int64, and finally from Int64 to Double. The values obtained are in meter and the angles are obtained in radian. The real problem arises when for the next updated position, the socket needs to be closed and opened once again in order to get the next position.

For continuous data acquisition, this process is entered in a while loop whose only exit condition is either when the interface is closed or when the cancel button is pressed. But if there is a continuous while loop, the resources are held up by this while loop and nothing else can be done during this time. There are two solutions to this problem. First, is initiating another thread parallel to the main thread and putting this while loop in this thread. Another solution is called a background worker in Visual Studio. A background worker is used when the program has some heavy calculations or lengthy operations to be done during the use of the interface. The background worker performs these operations in the background so that the active features of the interface such as buttons and textboxes can be used. When the calculations of the background worker are done, it can even report the progress and return the values once finished.

The time taken between two iterations of this cycle is approximately 0.2s to 0.3s. When there are more operations or when there are more background workers then this time can go up to half a second. Although this is not an optimal solution, there was no other possible method at the time when this Project was carried out.

Trajectory Plotting

The initial plan was to do the 3D-trajectory plotting of the robot in real time in Visual Studio. But Visual Studio allowed only plotting of 2D Data and no internal 3D Plotting support existed. Various third-party Application Programming Interfaces (APIs) were available online but none of them suited the application. Finally, it was decided to use an external software for 3D-Trajectory Plotting. The software for this application is called GNU Octave. GNU Octave is software featuring a high-level programming language, primarily intended for numerical computations. [6] This software is a freely redistributable software as well as one of the major free alternatives to MATLAB.

This software is called from the Interface in Visual Studio. Visual Studio saves the 3D-coordinate data that the robot has travelled in a "*.dat*" file which is used by the Octave code to make a 3D Plot.

The final plot formed by the program is shown in Figure 7. The point (0,0,0) is the base of the robot and the various points shown in the figure are the actual 3D-coordinates of the robot.

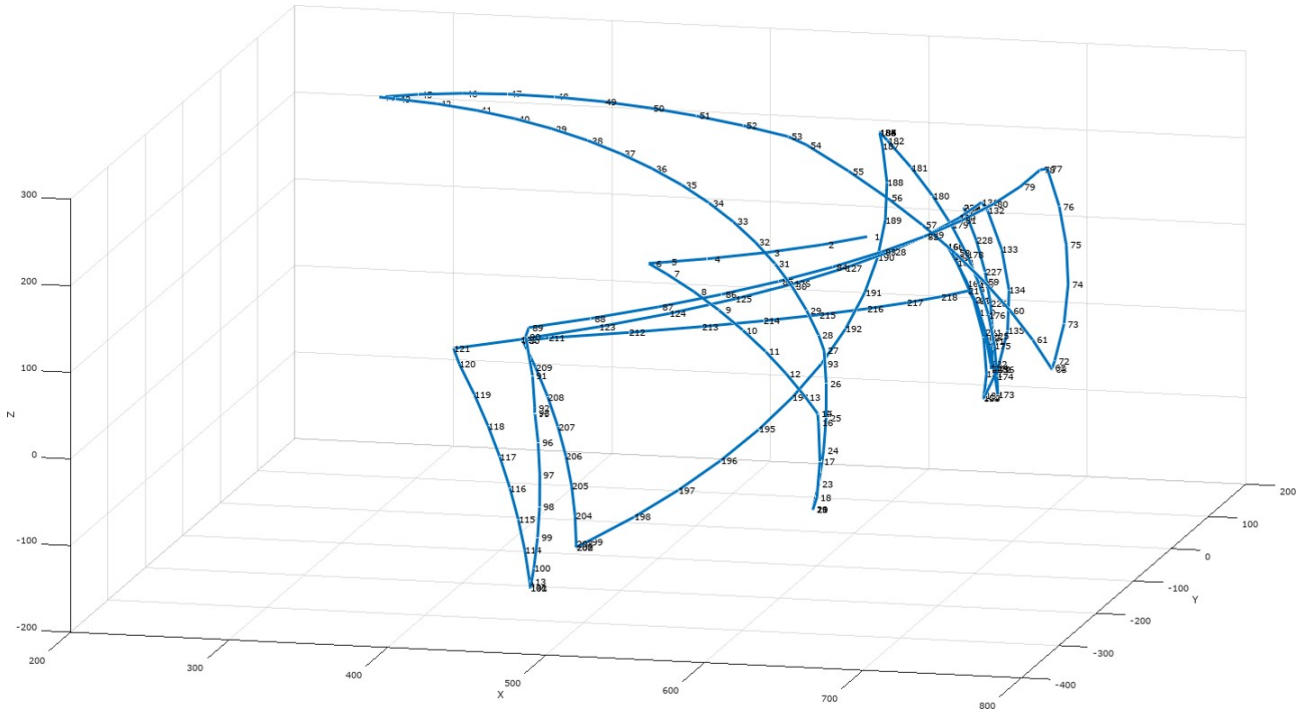

Figure 7: 3D-Trajectory plot of UR Robot in GNU Octave[14]

4 DEVELOPMENT OF DYNAMIC LINK LIBRARY (DLL) FILE

As previously described, the robot can be controlled externally through URScript Programming Language. This language contains all the commands needed to move and control the robot. These commands can be sent over TCP/IP Socket to the robot. The developed DLL file contains many of such commands required for movement of the robot and data acquisition.

The interface shown in the previous section uses functions which are programmed in this DLL file. Creating a separate file allows modularity in programming. This means that if another person also wants to develop a certain application using the UR Robot, they can easily integrate the developed DLL file in their IDE and prevent “reinventing the wheel”, which means they can focus on programming their application without worrying about the back end of the code.

[14] Own illustration

Functions of DLL File

The following explains all the functions in this DLL file.

1. **UR5Connect**

```
Socket UR5_DLL.UR5Connect(string ip, int nPort)
```

Connects to the robot.

Input: IP address as string, Port number as integer

Output: Connected socket

2. **UR5Disconnect**

```
Socket UR5_DLL.UR5Disconnect(Socket s)
```

Disconnect the program from the robot.

Input: Connected socket

Output: Disconnected socket

3. **Conv_deg_rad**

```
string UR5_DLL.conv_deg_rad(string s)
```

Converts degree value to radian.

Input: String

Output: String

4. **Conv_rad_deg**

```
string UR5_DLL.conv_rad_deg(string rad)
```

Converts radian to degree.

Input: String

Output: String

5. **Movej**

```
string UR5_DLL.movej(Socket s, string x, string y, string z, string rx, string ry, string rz)
```

Performs a joint arc move.

Input: Socket, (X, Y, Z, as mm, and Rx, Ry, Rz in radian, as a string)

Output: Command sent to the robot as string

6. **Movel**

```
string UR5_DLL.movel(Socket s, string x, string y, string z, string rx, string ry, string rz)
```

Performs a linear move.

Input: Socket, (X, Y, Z, as mm, and Rx, Ry, Rz in radian, as a string)

Output: Command sent to the robot as string

7. **Gripper**

```
void UR5_DLL.Gripper(Socket s, string width)
```

Opens and closes the gripper as per the width given in mm.

Input: Socket, width of gripper fingers in mm(1-105mm)

Output: None

8. **UR5_sdata**

```
void UR5_DLL.UR5_sdata(Socket s, string data)
```

Send command to the robot.

Input: Socket, data/command as string

Output: None

9. **UR5_get_pose**

```
string[] UR5_DLL.UR5_get_pose(Socket s)
```

Get the current position of the robot.

Input: Socket

Output: String array of size 6 for position data in the order: X, Y, Z, Rx, Ry, Rz

Note: UR Robot sends a package of 1044 bytes in which there is information about the current position of the robot. In order to get the updated position values everytime, disconnect and connect the Socket and then use this function.

10. **Move_Xplus, Move_Xminus, Move_Yplus, Move_Yminus, Move_Zplus, Move_Zminus**

Move in the selected axis with defined distance with defined velocity and acceleration.

```
bool UR5_DLL.MoveXplus(Socket s, double distance, double speed, double acc)
```

Input: Socket, distance in meter, velocity as m/s, acceleration as m/s^2

Output: Boolean

Note: Before using this function, always disconnect the socket and connect it again.

5 PROBLEMS

The problems faced during this Project are:

1. Due to current method to obtain position data, the rate at which the pose data is extracted from the complete incoming data is not efficient. Sometimes, this leads to a small delay when the robot reaches its initial goal position and it has to travel further to another position.
2. The gripper operation posed to be problematic at times and this could be a problem when using in a custom path definition. The gripper movement commands as shown in the gripper manual do not work. A new method was designed to send an entire gripper file every time after changing the value of gripper opening width in it. But this method is not efficient.
3. The robot does not have any external sensors and the only way of safety stop for this robot is to provide considerable force in opposite direction to the direction of motion. This can sometimes be dangerous as it requires a collision to stop. Various precautions had to be taken while programming this robot to avoid such collisions.

6 CONCLUSION

As robotics becomes more prominent in industry as well as research, the smother and easier operation of the robot also becomes a priority. This has been realized through development of the User Interface in this Project as a glimpse of easier usability of a robot. As IWT Wirtschaft und Technik GmbH works closely with DHBW Ravensburg, and more students perform their Student projects with robotics, the developed DLL file aims to make the work of future students easier, who want to work with any robot from Universal Robot. Although this Project implements an UR5 Robot, the UI and DLL can also be used for other Universal Robots variants. As the base of the UI is ready, it also motivates the students who work with robots from other manufacturers and continue this development.

The DLL file contains the functions required to program the robot for any required application with UR Robot. The UI can be easily used by anyone, to make basic movements or even to define custom paths along with picking up or dropping an object.

REFERENCES

[1] International Federation of Robotics, *Industrial Robots: Robot Investment Reaches Record 16.5 billion USD*, 2019. Accessed: Apr. 21 2020. [Online]. Available: https://ifr.org/ifr-press-releases/news/robot-investment-reaches-record-16.5-billion-usd

[2] Microsoft, *Visual Studio.* Accessed: Apr. 27 2020. [Online]. Available: https://visualstudio.microsoft.com/vs/

[3] Universal Robots A/S, *UR5 Technical Specifications.* [Online]. Available: https://www.universal-robots.com/media/50588/ur5_en.pdf (accessed: Apr. 21 2020).

[4] *Robots and robotic devices — Safety requirements for industrial robots — Part 1: Robots,* ISO 10218-1:2011, 2011.

[5] OnRobot A/S, *RG2 Datasheet.* [Online]. Available: https://onrobot.com/sites/default/files/documents/Datasheet_RG2_20191122.pdf (accessed: Apr. 21 2020).

[6] GNU, *Octave.* Accessed: Apr. 27 2020. [Online]. Available: https://www.gnu.org/software/octave/about.html

A Setup using Cobots for Training both CPS and Sensorics in Engineering Education.

Prof. Dr.-Ing. Thomas Dietmüller[15], Prof. Dr.-Ing. Lars Ruhbach[16], B.Eng. Niklas Hohenauer[17], B.Eng. Matthias Münsch[18]

Keywords:
Robots, Collaborative Robots, Cobot, Education, Sensorics

[15] DHBW Ravensburg Campus Friedrichshafen, Professor Maschinenbau
[16] DHBW Ravensburg Campus Friedrichshafen, Studiengangsleiter Maschinenbau; CEO IWT Wirtschaft und Technik GmbH
[17] Absolvent der DHBW Ravensburg im Studiengang Maschinenbau
[18] Absolvent der DHBW Ravensburg im Studiengang Maschinenbau

1 INTRODUCTION

Digitalization is a major challenge as well as a great chance for German machinery and plant engineering. Accordingly, competences of engineers and requirements on qualification need to change to master digital transformation in production.

Although basic engineering knowledge remains key, several specialties such as automation, sensorics or robotics become more important. Those subjects have been strengthened in newer engineering curricular. Additional subjects like cyber-physical systems (CPS) or smart production had been added to bachelor engineering education. In terms of methodological competences future engineers need to be good process and systems thinkers [1]. Not to mention – the widely agreed and unquestioned - ability to design and conduct experiments to identify, formulate and solve engineering problems as being most important [2]. However, problem-based learning should be based on authentic and also more complex problems to be successful [3].
Thus, in up to date engineering education all this needs to be addressed. Fortunately, some of those subjects are strongly related, not only technical such as robotics and sensorics: Educators recognized that students performing robotic tasks are strongly engaged and dedicate extra time to solve these problems. The value of using educational robotics lies in engaging students to apply their knowledge of different subjects to solve real-life problems [4].

With this in mind it seems obvious to use real-world automation problems to come up with a robotics solution and train subjects like sensorics or CPS in engineering education. In this paper we present an approach using a cobot and a visual sensor as part of a laboratory exercise for mechanical engineers. In an industry like scenario, students learn to set up and use a basic CPS to automate decision-making on the quality of components in production.
Before explaining the educational setup, this paper gives a short introduction to CPS, machine-to-machine (M2M) communication and robotics.

2 CYBER-PHYSICAL SYSTEMS (CPS)

CPS are objects, gadgets, systems and logistic components that directly connect information technology and software with mechanical and electronic components. Using sensors, a CPS can gather information about its surroundings. A microcontroller then evaluates the gathered information and controls the connected actors accordingly [5].

However, CPS can be more than plain control loops. They break up the rigid connection between sensors and actors and enable the system to make situational decisions based on given priorities and external inputs. In that case, the same input does not always cause the same output. Furthermore, artificial intelligence gives CPSs additional functions. They can adjust their decisions in favour of common goals and learn how to deal with different situations as kind of swarm intelligence. This allows CPS to work almost autonomously, once they have been programmed [6]. In order to align their goals, the CPSs must communicate with each other. The direct communication is another key factor in the recent success of CPS. It is possible, because every CPS has the means to process and provide data. Thus, there is no need for additional communication layers. For that reason, the modern network topology moves away from the typical pyramid scheme towards an interconnected network of CPS in automation.

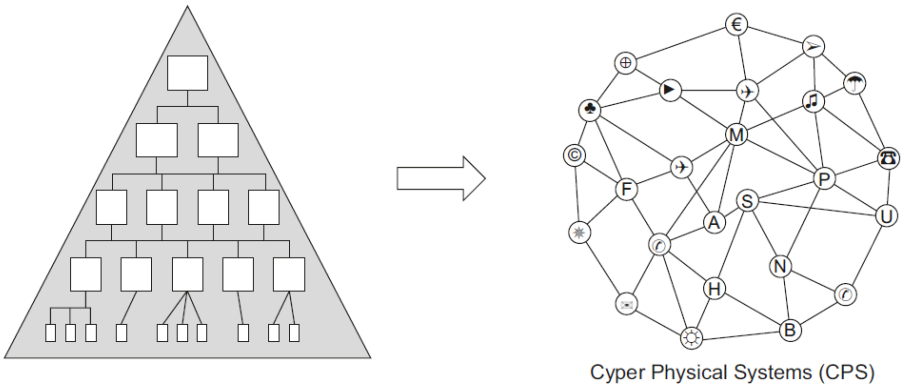

Figure 8: Pyramid scheme and interconnected networks [7]

However, the main advantage of interconnected networks is their high flexibility and self-adaption. Individual CPSs can be added, exchanged or removed at will, because every system in the network is an independent unit, that could work completely on its own. As long as no important data access points are altered, the rest of the network should not be affected by the change. Additionally, each individual CPS has a high computing capacity. Combined with the direct communication between all of them, the network has real time capability.

An exemplary application for such a real time communication is the educational setup. In this setup, a sensor provides real time measurement data to a robot. Applications like this are becoming more and more common in the modern industry.

2.1 MACHINE-TO-MACHINE (M2M) COMMUNICATION

M2M communication refers to the communication between computers, embedded systems, mobile terminals, sensors and actuators with little to no human intervention [8]. Essentially, it is the communication in CPS.

Industrial M2M communication can be implemented through both, wired and wireless connections. While wired connections are more reliable, wireless connections are more flexible and obviously better suited for mobile devices.

CPS, that can communicate in a network offer more opportunities for applications. Because of that, they are generally more valuable than standalone systems. For this reason, there is a tremendous increase in the annual data volume of M2M communication.

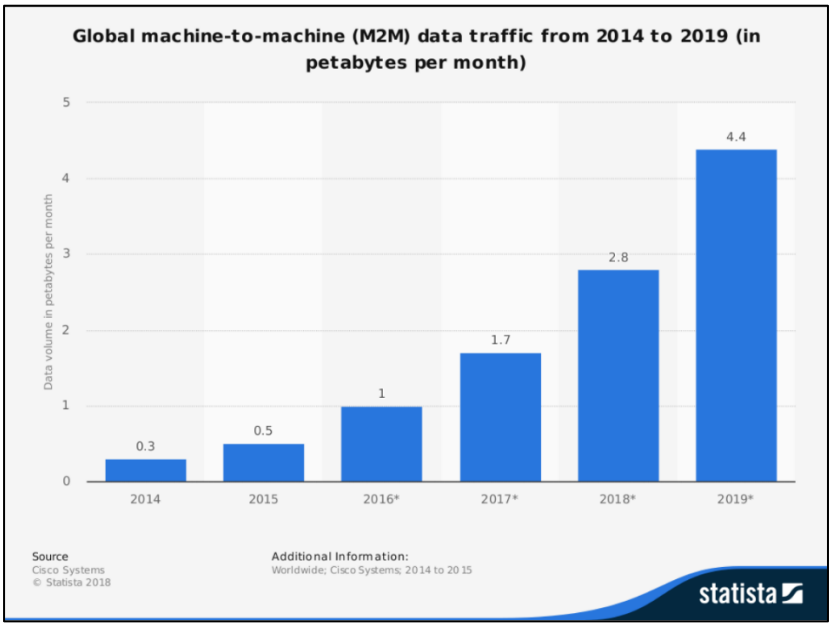

Figure 9: Expected annual data volume in M2M communication [9]

Figure 2 shows the expected M2M communication data volume in the worldwide mobile communications network, measured in petabytes per month. This drastic increase shows the importance of M2M communication in the modern world. The M2M communication in industry will experiences a similar increase. Therefore, it is important to educate students on CPS and M2M communication. The educational setup utilizes a simple communication protocol.

Because there are two participants in the network, it is easier to wire them both together and control the communication through their individual programs. This basic form of M2M communication is more comprehensible for the students working on the setup.

However, bigger CPS networks require a uniform communication protocol, that provides flexibility and platform independence. The OPC unified architecture is an example for such a communication protocol [10].

2.2 AUTOMATED DECISION MAKING

The amount of data generated in a modern smart factory is ever increasing. As a side effect, there are also more decisions to be made. With the huge data volume at hand, humans alone will not be able to make efficient decisions in the necessary time periods [11]. Thus, it is important to automate as many decisions as feasible.

The educational setup shows an application, where the decision, whether a part gets accepted or rejected, is made automatically. This decision is made by the sensor, integrated into the setup. Before the first measurements can be made, a master-part is loaded into the sensor. After that, it can start measuring. Every new component, the robot inserts into the sensor will be compared to the master. If differences are within the previously programmed tolerances, the component gets accepted. Otherwise it will be rejected. The sensor communicates its decision to the robot, which sorts the part into the corresponding box. With this application, students learn the basics of automated decision making.

3 DIGITALIZATION COMPATIBLE HARDWARE IN PRODUCTION

In modern productions, hardware, which is suitable for the digitalization, is becoming increasingly important. Those systems are characterized by the consistent integration und the networking of all production resources [12]. In the sense of the smart factory, the physical and digital technologies should fuse into CPS. The main part in this process are robots and not just to move components or support employees. Robots will also be further integrated into quality control [13].

Cause of the rapid increasing range of models the area of application for robots gets larger and larger. In nearly every company size, a flexible automation of manufacturing processes will be possible, because of the wide range of applications of robots.

The flexible automation ensures that the individual customer requirement and a flexible production can be fulfilled. In times of speed up business cycles the point before is an important advantage and paved the way for the advance of the industrial robots [14].

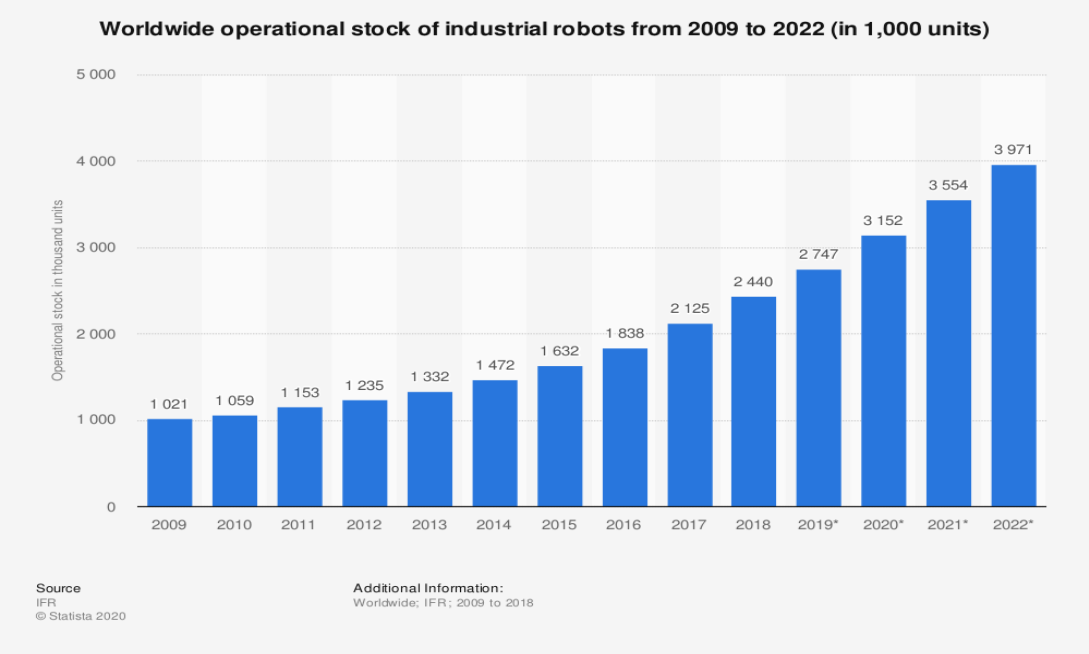

Figure 3 Statistics stock of robots worldwide [15]

Figure 3 shows the increase of the worldwide stock of robots. For this reason, it supports the points mentioned above that robots are becoming increasingly essential in modern production. They had been and will stay a key element for higher productivity in production. The different types of robots and digitalization-compatible sensors for measurement are explained below using the educational setup.

3.1 TYPES OF ROBOTS

These days there are different types of robots on the market. On the one hand this includes service robots, which do daily work like mow the lawn or hoover for us. On the other hand, medical, free time, humanoid and bio robots are a part of our lives [16]. However, these robots are not widely used in the production of modern manufacturing companies.

In today's production are mainly industry robots used. They are built up out of a robot arm, which is used to take over uncomfortable and heavy work without any rest. This work is mostly not mentally demanding. Out of this, industry robots do tasks like transporting of material and the onloading of industry plants [17]. But joining activities such as welding, gluing, riveting or screwing have long been part in the work day of an industrial robot.

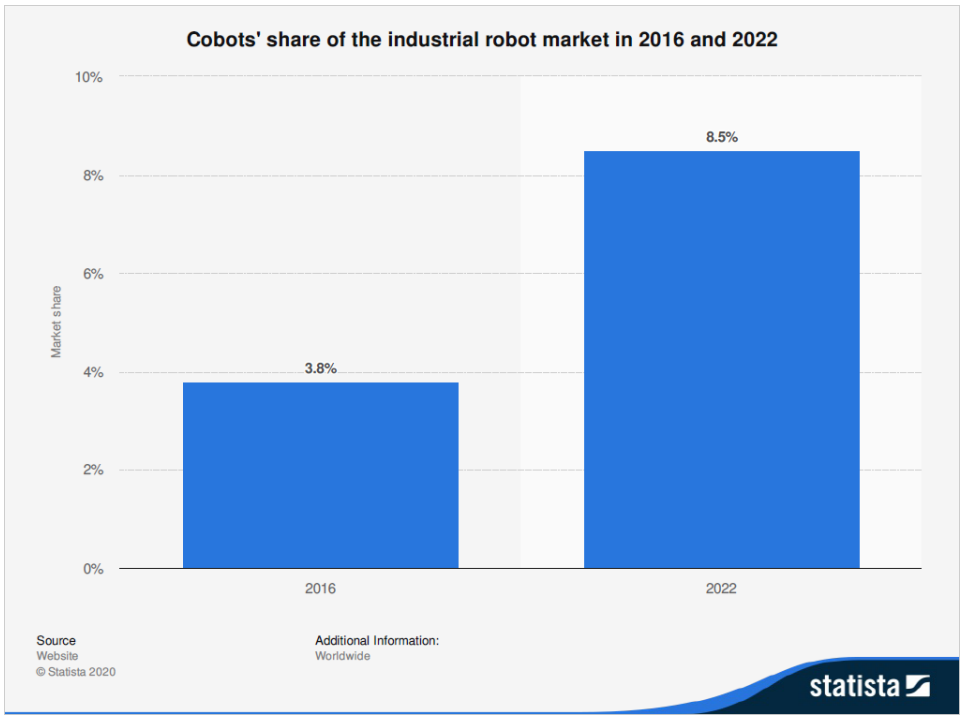

Figure 4 Cobots` share of the industrial robot market [18]

One rather young type of industrial robots is growing particularly strong according to Figure 4. These robots are the collaborative robots or cobots. They are characterized by their low acquisition costs and low inertia. Cobots have flexible uses, because of their low weight they can be easily transported. That is a really important point for a flexible production. In addition, these robots are suitable for collaborative work with humans. Every type of cobot can be operated without any safety fence, if all safety regulations are complied. Since the cobots have a low working load, they are mainly used for assembly [19].

4 EDUCATIONAL SETUP

After the explanation of the state of the art we return to our educational setup. In the following the implementation of the educational setup is described from planning to implementation and finally the first trainings. In this setup, students are to automatically measure components by using a cobot and a visual sensor. In addition, the robot should sort the components based on the decision of the sensor, which differentiates between the measured values and the entered limit values between good and rejected components.

The idea for the educational setup has its origin in real-life production. In manufacturing of high precise turned components, it is usually required to conduct 100% dimensional test to ensure that every component meets the specification. Normally this is done manually.

However, the manual testing is likely to results in errors. Additionally, the testing is likely not to be done in sequence with the production – with the result of late feedback to machining. Thus the automation of the manual testing could improve both quality and costs.
The educational setup uses this measurement scenario to motivate the problem-based learning process.

4.1 PLANNING

The development and implementation of the educational setup was done as a project itself. At first a project discussion was conducted to clarify requirements and define the project management method. To both work structured as well fast we decided to use hybrid project management to create the setup.

After a rough planning of the time schedule, the different tasks were split in the group. The tasks were in the way, that every group member saw into the software and the hardware parts of the project. So, everyone had the chance to bring in ideas. For the documentation of the task splitting the organisation app “Trello” was used. This also brought the advantage, that the planning could be done on smartphones anywhere and anytime. With the app it was also simple to review and adapt the backlog and the tasks in process. It also was possible to share and deposit the necessary documents for the setup.

After familiarization, the entire setup with its necessary components was designed (see figure 5). The last point of the planning included the inquiry of the necessary components and the procurement of these. At this on the one hand, it is important to control the delivery time to be able to meet deadlines.

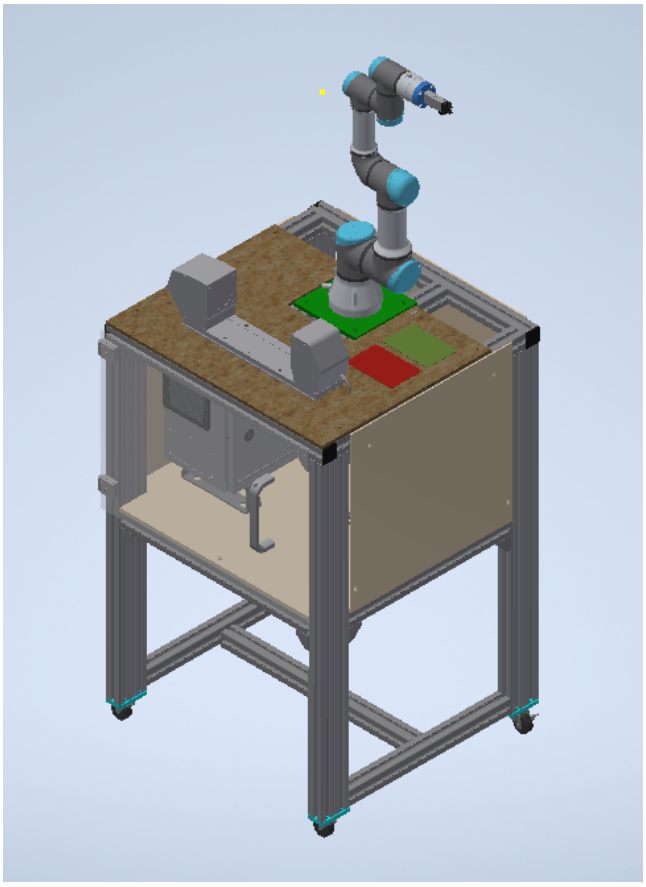

Figure 5: CAD-model of the setup

4.2 Implementation

After finishing the development phase and the 3D-model, it was time to build the setup. First, a parts list was derived from the 3D-model. The needed parts were then procured through different means. Most of the screws and some aluminium profiles where already available at the campus' workshop. The rest of the profiles and the necessary attachments, like connectors, wheels and hinges for the door, were ordered online. Additionally, wooden and plastic panels for top, bottom and side-covers were ordered. These panels came pre-cut, so no additional machining was needed before the assembly. Further, a 12 mm aluminium plate was ordered to serve as the robot's attachment. Both the adapter and the fingers for the cobot's gripper were 3D-printed, so the only things left to order were the two boxes for the good and the reject components.

The assembly of the setup required basic machining knowledge, as the profiles had to be cut to length and a lot of holes for the connectors had to be drilled. However, with a bit of patience, it was rather easy to assemble the parts. The cobot was fixed on two rails. Because of that, it's position on the station can be varied. This keeps the setup flexible, so it could be used in other applications, too.
Because the whole station stands on tires, it can easily be moved, further adding to its flexibility. The control units, the power adapters and the excess cable length are placed inside the stations compartment. The compartment can be accessed through a door. The following figure 6 shows the finished setup.

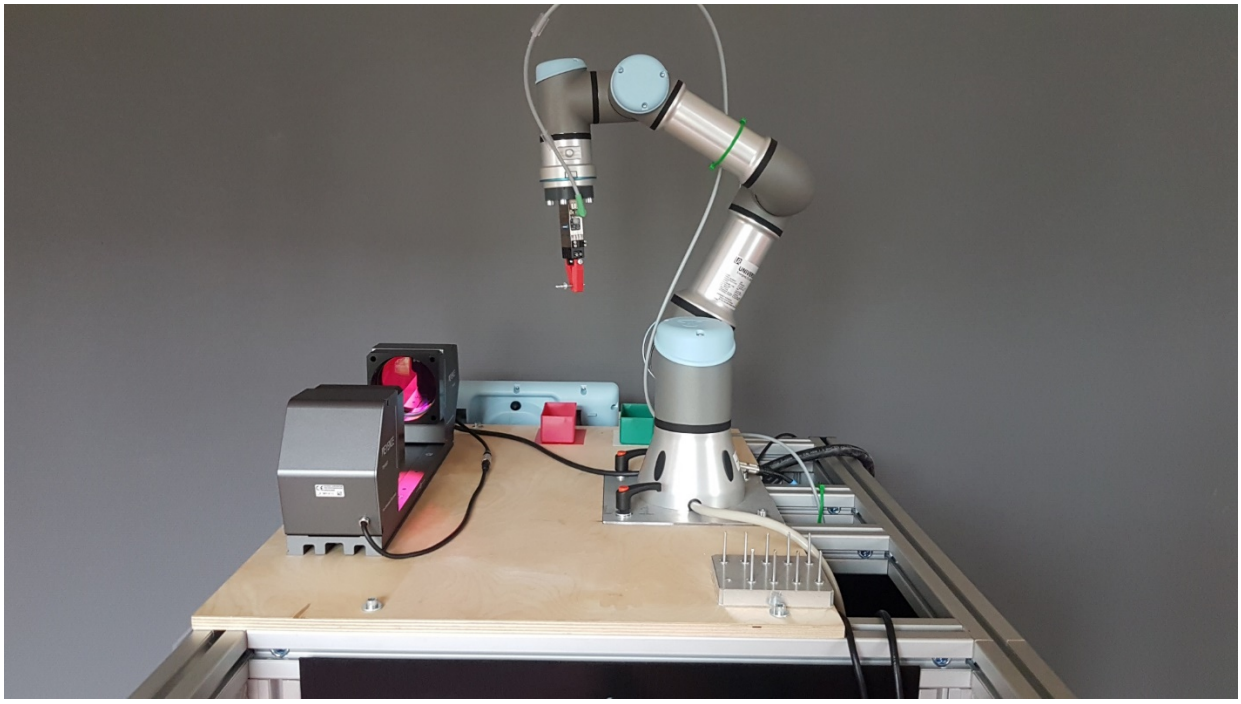

Figure 6: Mobile station with robot and measuring device

4.2.1 PROGRAMMING THE COBOT

The next step, after finishing with the assembly and wiring of the hardware, was to implement the programs for both, the cobot and the visual sensor.
Learning the basics of programming an UR3 cobot was rather simple, due to Universal Robots good and free online training. This beginner's tutorial can be finished in two hours and is very helpful. It covers six modules ranging from a first look to interacting with external devices and safety settings [20]. The cobot's program was written with the knowledge from the online training, coupled with "learning by doing". Later the same online training will be used by the students to get the cobot working correctly in the laboratory exercise.

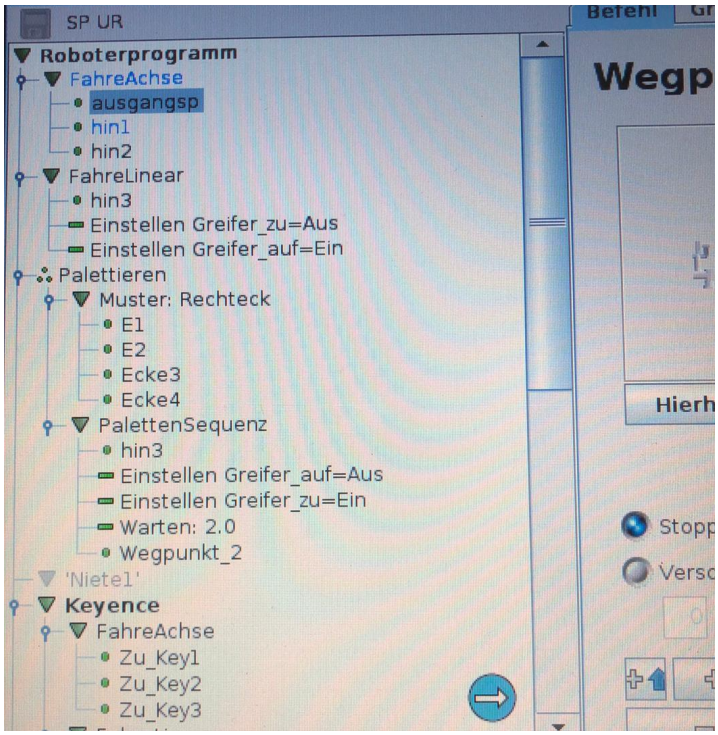

Figure 4: programming the robot

The written program includes three basic tasks. Firstly, the cobot picks up the part to be measured from a defined position and inserts it into the visual sensor's field of view. Then, it waits for the sensor's response, whether the part will be accepted or rejected. After it receives the signal, it sorts the part into the corresponding box. This process continues, until all components have been measured and sorted. From this simple program, students can learn how to control a gripper, communicate with external devices, implement conditional steps and how to move the cobot to defined waypoints.

4.2.2 SETTING UP THE VISUAL SENSOR

Keyence 2D-Micrometer comes with a detailed user's manual [21], which describes how to load a master-part, define measurement areas, adapt tolerances and configure angle and position compensations. The software is very graphical. Thus, it is easy to get started with programming.

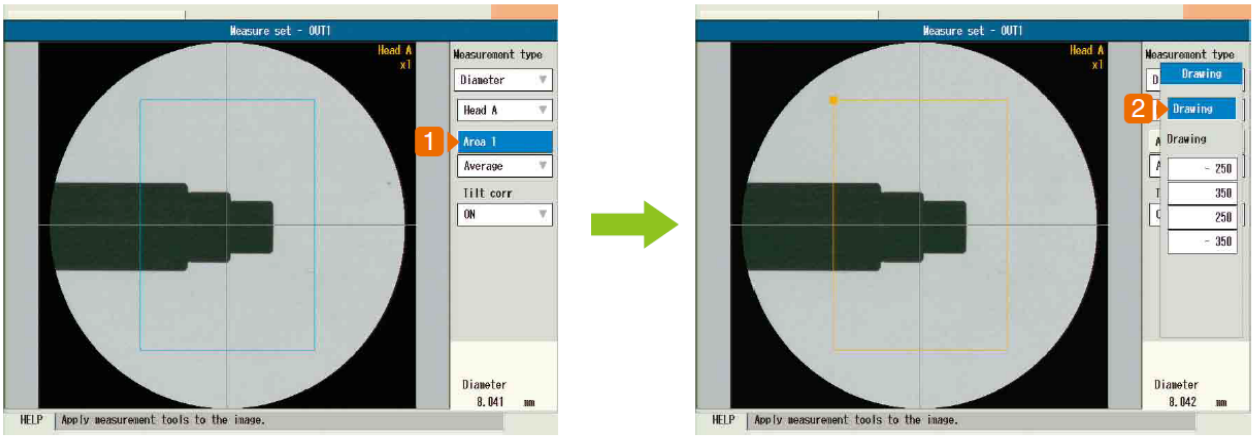

Figure 5: Setting the measurement area [22]

Measuring areas can be defined by simply marking the areas with boxes, as shown in the picture above. Then the program just need's the additional information, of what to measure in said area. For example, it could measure the maximum diameter inside the area. After defining tolerances, the first measuring point is set. For the educational setup, a few measuring points on our components angles, diameters and lengths have been prepared. The program 's main purpose is to show the different possibilities the visual sensor offers. This allows students to get a grasp on the scope of 2D optical metrology. Furthermore, the software is easy to use, so they can even get involved in implementing a simple measuring program through the exercise mentioned below.

4.2.3 PROGRAMMING THE INTERFACE

The interface between the cobot and the visual sensor consists of four data lines. These data lines have been hardwired. They connect the control units of both devices. Each of the cables carries a binary signal. The whole interface is implemented into the cobot's program. Thus, it reads the visual sensor's outputs and writes its input. The four signals required to run the program are:

Ready_A This is one of the sensor's outputs. It tells the cobot when a measurement can be started.

Trigger_A This is the only one of the sensor's inputs, that is needed for the communication. The cobot uses it to trigger a measurement, after the sensor is ready and a part has been inserted.

Busy For the time the visual sensor is conducting the measurement, this signal stays on. When it turns back off, the cobot knows that the measurement has been finished and the result is on the "GO" data line.

GO This signal only turns on, if the measurement results are inside the given tolerances. Therefore, it tells the cobot whether to accept or reject the measured part.

These four signals provide enough data, to realise the required communication in the CPS, while keeping the whole setup simple and comprehensible.

4.3 DEVELOPMENT OF A SUITABLE EXERCISE AND PROVIDING IT FOR STUDENTS

First, appropriate components to be used in the laboratory exercise were chosen according to selected setup hardware. Because of the specifications of the 2D sensor the components should be rotationally symmetric. Additionally, the classification data of the sensor, the robot and the gripper should be involved.
Secondly, an online course was created. It explains the educational setup's basic ideas. For this the open-source learning platform Moodle was used. The following information had been provided via Moodle online platform:

- important key data for the setup of the exercise,
- the mode of operation of cobot and sensor and its programming,
- the communication between the cobot and the visual sensor

The students are required to work through these documents before the practical exercise takes place. At the end of this preparing they must complete a test. The test is used to query the most important things and to memorise them. Altogether the students will roughly need one hour for their preparation.

The documents for the practical part of the exercise have also been created und uploaded in Moodle. The first document describes a scenario in a manufacturing company matching the educational setup. This made up story is supposed to be compelling and interesting for the students, while also guiding them through the practical exercise, as they must solve several problems that could occur in a real production environment. The documentation provides a step by step guide on how the problem solving could be done. The target of this exercise is that the students work independently in their groups and apply the knowledge they gathered from the exercise's theory part to successfully solve the company`s problem.

To put the setup into operation the students will find themselves faced with the following problems:

1. First, the interface between the cobot and the visual sensor needs to be rewired: The students learn the basics of wiring, as well as which wires are needed for the communication.
2. Next, the gripper has to be attached to the cobot's flanch. Additionally, the data cable has to be attached to the robots control unit: The students learn, how cobots or robots in general control their grippers. Furthermore, they should learn that, robots often need different adapters to attach the right grippers.
3. Then, the Missing measurement points must be programmed into the visual sensor, using an existing master-picture: The students learn the basics of working with a typical optical measuring software used in automation industry.
4. After that, the missing waypoints for the good and the reject components must be taught to the cobot: The students learn basic programming of a popular cobot.
5. Finally, an if-instruction must be implemented into the cobot's program, so it throws the measured part into the right box, according to the input of the visual sensor: The students learn about conditional instructions and the basic programming of interfaces in CPS.

Students roughly need 2-3 hours to solve the practical problem and get the setup successfully running. In case they encounter any difficulties, while working on the given tasks, the written instruction includes additional clues. If that is still not enough help, the instructor will support the students. Eventually the students have implemented a cobot doing automated visual inspections and as a result improved their understanding of CPS.

5 CONCLUSION

Overall, the educational setup provides a great opportunity for students for problem based learning in an authentic and ambitious scenario. Thus, preparing engineering students for problems, they might actually face in their near future. Moreover, it serves as a practical example to go with the theoretical knowledge, students receive in their lecture, while also diving into further detail.

First groups of students have already successfully completed the exercise. So far, the feedback is very positive. They enjoyed the change from their everyday studies, as well as the „learning by doing" approach, the exercise takes. Soon, more students will get to work on the exercise.

In order to further improve and enlarge the setup additional features could be implemented. One way to upgrade the setup could be implementing Robot Operating System (ROS). This is an open source software that provides hardware abstraction, device drivers and other functions [23]. Another possibility in terms of communication is the implementation of the OPC-UA protocol, as mentioned in chapter 2.1. Another way could be to make the picking process more challenging by providing the components as bulk. The students then would have to use an additional camera and software to solve the bin picking scenario.

In summary, the setup combines educational robotics and problem based learning to better understand and expand practical knowledge on robotics, sensors and CPS in engineering education.

6 REFERENCES

[1] G. Spöttel, L. Windelband: Industrie 4.0, Risiken und Chancen für die Berufsbildung, 2. Auflage, wbv Publikation, 2019.

[2] E. Koehn: Engineering perceptions of ABET accreditation criteria Journal of Professional Issues in Engineering Education & Practice, pp. 123, 1997.

[3] D. Jonassen, J. Stobel: Everyday Problem Solving in Engineering: Lessons for Engineering Educators, Journal of Engineering Education (JEE), 2005.

[4] J. Johnson: Children, robotics, and education. Artif. Life Robot.7(1), pp. 16–21, 2003.

[5] S. Luber: Was ist ein Cyber-physisches System (CPS)?, [Online] Available: https://www.bigdata-insider.de/was-ist-ein-cyber-physisches-system-cps-a-668494, Accessed on: April. 26 2020.

[6,7] F. Hüning: Embedded Systems für IoT, Springer Vieweg, Berlin, 2019

[8] M. Chen, D. Li, J. Wan, F. Xia, K. Zhou: From Machine-to-Machine Communications towards Cyber-Physical Systems, 10. Auflage, ComSIS, 2013.

[9] statista, [Online] Available: https://www.statista.com/statistics/267310/data-volume-of-non-internet-ip-traffic-by-category/ Accessed on: April. 24 2020.

[10] OPC Unified Architecture, [Online] Available: https://opcfoundation.org/about/opc-technologies/opc-ua/, 2020, Accessed on: April 04.2020.

[11] K. Lucks: Praxishandbuch Industrie 4.0, Branchen – Unternehmen – M & A, Schäffer-Poeschel Verlag, Stuttgart, 2017.

[12] A. Knoll: Traditionelle Industrieroboter ohne Schutzzaun, Markt & Technik, (4/2020), 2020.

[13] A. Knoll: Roboter als Qualitätssicherungs-Tools, Markt & Technik, (04/2020), 2020

[14] A. Burkert: Roboter übernehmen die Welt der Produktion, [Online] Available: https://www.springerprofessional.de/industrieroboter/industrie-4-0/roboter-uebernehmen-die-welt-der-produktion/15236792, Accessed on: April 04.2020.

[15] C. Müller: Worldwide operational stock of industrial robots from 2009 to 2022, International Federation of Robotics IFR., [Online] Available: https://ifr.org/downloads/press2018/IFR%20World%20Robotics%20Presentation%20-%2018%20Sept%202019.pdf, Accessed on: April 24.2020.

[16,17] M. Haun, M.: Handbuch Robotik, Springer Verlag, Berlin Heidelberg, 2013.

[18] Industrial-Robots-Infographic, [Online] Available: https://www.interactanalysis.com/wp-content/uploads/2018/07/Industrial-Robots-Infographic-image.png, Accessed on: April. 05 2020.

[19] W. Bauer, M. Bender, M. Braun, P. Rally, O. Scholtz: Leichtbauroboter in der manuellen Montage – einfach EINFACH anfangen, Frauenhofer IAO, Stuttgart, 2016.

[20] V. Charbonneau: Learn to Operate and Program Cobots Through Free Online Training, [Online] Available: https://www.engineering.com/Education/EducationArticles/ArticleID/14363/Learn-to-Operate-and-Program-Cobots-Through-Free-Online-Training.aspx, Accessed on: April. 10 2020.

[21,22] Keyence: 2D-Mikrometer Modellreihe TM3000, Benutzerhandbuch, Keyence Corporation, Osaka, 2019.

[23] ROS Documentation, [Online] Available: http://wiki.ros.org/, 2018, Accessed on: April. 25 2020.

Herausgeber

Prof. Dr. Heinz-Leo Dudek
Prorektor und Dekan der Fakultät für Technik

Duale Hochschule Baden-Württemberg Ravensburg
Baden-Wuerttemberg Cooperative State University
Marienplatz 2
88212 Ravensburg

ISBN 978-3-945557-01-3

ISSN 2199-238X

DOI 10.12903/DHBW_RV_FN_01_2021_SHETH_DALM_SAHUJI_DIETMUELLER_RUHBACH_HOHENAUER_MUENSCH