Easy Programming module - Integrator version

Project Title	**Digital Technologies, Advanced Robotics and increased Cyber-security for Agile Production in Future European Manufacturing Ecosystems**
Project Abbreviation	**TRINITY**
Project Funding Scheme	**H2020 Innovation Action (IA)**
Call Identifier	**DT-ICT-02-2018: Robotics - Digital Innovation Hubs (DIH)**
Project Website	**http://www.trinityrobotics.eu/**
Project Start Date	**1.1.2019**
Authors	**Flanders Make, Ali Bin Junaid, Robotics Application Engineer**

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

1 Contents

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

2 Training module overview

This training module will provide a technical overview on the 'Easy programming module'. After reading this document, you should be able to:

- start the application on the main computer
- run the code on the devices side
- open the easy programming web interface
- start programming via this interface

The tutorial will be split in different parts. A first reminder of the hardware material will be done. Then, the tutorial will first focus the framework side which will take longer to setup. The framework needs to be able to correctly interact with external devices. In this module, the hardware devices are the KUKA IIWA collaborative robot and the Robotiq 3 fingers gripper for the gripper attached on the cobot.

In order to correctly setup the framework, you need to have access to the source code. If you don't have access to it, please contact Flanders Make for files access.

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

3 Framework – technical overview

Before starting to build the code for the easy programming framework, requirements first need to be met. As explained during the use-case and module videos, a computer running a Linux distribution is required. The demonstration was performed using Ubuntu 16.04 LTS. Using a newer Ubuntu distribution such as 18.04 or 20.04 should still be compatible. However, this was not tested.

3.1 How to setup the component of system (framework)

The complete framework code is stored inside a Gitlab repository. The first step in order to start the configuration of this module is to install git on Linux. Git is used here in order to access the source code from the content repository. If you have access to the source code, there is no need to install git. However, it is still important to update your Linux system.

For non-experienced in Linux user, you first need to open a command prompt (terminal) which allows you to install/setup the environment required to run the application. To open a terminal, you can either search for 'Terminal' in the applications list or you can use the shortcut "Ctrl+Alt+T".

Make sure your Ubuntu environment is up to date. To perform this, you first need to update and upgrade your system. Type the following lines inside the command prompt

```
sudo apt update
sudo apt upgrade
```

Inside this terminal, you can install git with the following command (copy/paste in terminal)

```
sudo apt install git
```

If you have no experience in Linux, sudo means that you execute the command as root (with administrative right), therefore requiring you to enter your password. The prompt will ask you if you want to install git. You need to enter 'y' in order to validate the installation.

In order to help the redeployment of this application onto a new computer, Docker was used in this project. Using Docker will save a lot of time in this integration phase. The module video shortly explained the advantages of Docker. In short, with Docker, you will not need to manually install the dependency requirements and spend a long time debugging in order to match all the different environment settings. The knowledge of this environment requirements is stored inside a Dockerfile.

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

You need to manually install docker and docker-compose in order to use the main functionality of Docker. Type down the following 2 lines in the terminal in order to install Docker

```
sudo apt install docker
sudo apt install docker-compose
```

The three packages that you installed are the only native requirements that you need to install on your PC. The other dependencies will be handled inside the Dockerfile.

3.2 Setup the environment (framework)

With those requirements met, you can now pull the code from the Gitlab repository (download the source code for the application). Navigate inside the folder where to want to store your native code. You can create a new folder with the following command

```
mkdir new_folder_name
```

Clone the repository inside the preferred folder with

```
git clone address_repository
```

(Contact Flanders Make for files access trinity@flandersmake.be). If you already have the source code, then you can directly move your source code inside the new folder new_folder_name

Quickly check that when you navigate inside the downloaded repository, you can find the source code (non-empty folder)

3.3 Build the application (framework)

If your downloaded folder is non-empty, you should be able to see a Dockerfile file. This file contains all the environment requirements. In order to use those requirements/libraries, the workflow is the following:

- Build an image based on the Dockerfile
- Run a container with this new Docker image
- Compile the application inside the container
- Run the container (either inside the container or outside)

Copy and paste the following lines in the command prompt in order to properly setup the application.

Navigate inside repository. If you type ls you should be able to see the Dockerfile. If you can find it then you can build the Docker image with the following line.

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

```
docker build . -t ros2
```

If you have permission error (permission denied), type the following line

```
sudo usermod -a -G docker $USER
```

and reboot your system.

The build of the image will be quite long. It installs ROS2 and the library requirements.

Once the image is built, you should be able see your built Docker image (ros2) with the following line.

```
docker image ls
```

The next step is to navigate inside the Docker container. For this you need to be careful about where your execute your line. It should be the same as for the docker build command.

```
docker run -it -v $(pwd):/home/ros2_ws ros2 bash
```

This file will launch the docker container. When inside, you are in a virtual environment where all the library/dependency requirements are installed. Once launched, you should be in the /home/ros2_ws folder. When you type ls, you should see the same file as before (Dockerfile, …).

The ros2 part of the previous command line means that the built Docker image ros2 will be used for running this container. The -v tag means that the local folder $(pwd) will be mapped to the virtual folder /home/ros2_ws. If you perform a change to the local file, the virtual file will also be modified (due to the -v tag).

Now that you are in the Docker container, with the right files being present in the container and with the right Docker image containing the dependency requirements, you are ready to compile the code. For this a bash file was created. You can execute the bash file with the following command.

```
bash build_new_messages.sh
```

This bash file will perform 2 important steps:

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

1. Build the ROS2 packages which are mandatory for this application (colcon build)
2. Build the web dependencies (npm install). This is necessary in order to use the web modules.

After the build, you should now see 2 new folders inside the main folder: install, build

After the built is complete, you can exit the container with the following line

```
exit
```

3.4 Run the application (framework)

There are 2 options if you want to start the application. The first option consists of entering the Docker container (same as previous section) and individually roslaunch all the required ROS2 nodes and database via the adequate command. The second option is to use docker-compose which allows you to launch a compiled code without requiring to be in the Docker container. Docker compose is the preferred choice as it allows you to launch the entire application (multiple containers with different codes running) with a specific configuration which includes the Doker image, volume as well as network.

Docker compose requires a specific configuration file: docker-compose.yml

You can launch the ROS2 components of the easy programming module with the following line inside the main folder (if you type ls, you should see the Dockerfile and docker-compose.yml)

```
docker-compose up
```

In another terminal, we will launch the database. To do this, you need to run the following line

```
. run_database.sh
```

Once, the 2 modules are launched, then you can open a web browser and type the following URL in order to access web easy interface

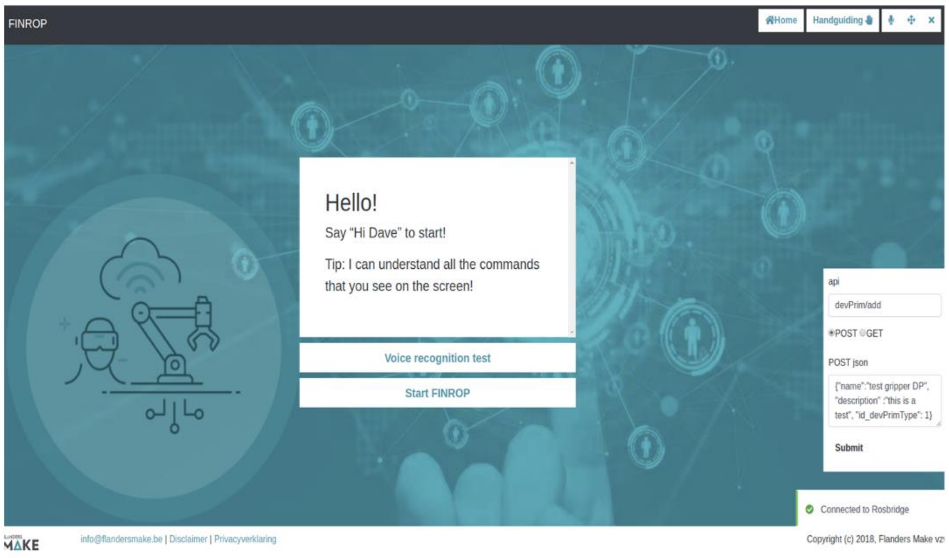
http://localhost:3000/html/app.html

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

You have launched all the required codes on the easy programming side.

3.5 Use web page

If you have correctly launched everything and you entered the URL, you should be able to see the following web page. This is the main page of the easy programming interface. To continue, you can click the ‘Start’ button.



Hello!

Say “Hi Dave” to start!

Tip: I can understand all the commands that you see on the screen!

api

devPrim/add

[✓ POST] GET

POST json

```
{"name":"test gripper DP", "description" :"this is a test", "id_devPrimType": 1}
```

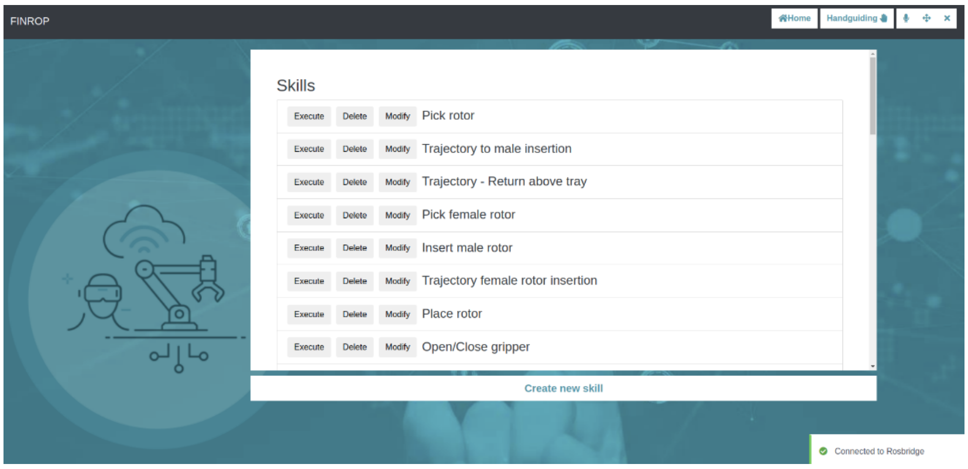
Connected to Rosbridge

info@flandersmake.be | Disclaimer | Privacyverklaring

Copyright (c) 2018, Flanders Make vz

You have now entered in the main application. At this stage you can see the instanciated skill list that are already present in the local database. For each of those skill, you have 3 options: execute, delete or modify. In the first case, the skill will be executed as a sequence of primitives steps that are included in the skill. The second option allows you to delete the skill if not relevant. The final option is to modify the skill.

Trinity project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No. 825196.



Skills

- Pick rotor
- Trajectory to male insertion
- Trajectory - Return above tray
- Pick female rotor
- Insert male rotor
- Trajectory female rotor insertion
- Place rotor
- Open/Close gripper

A final possibilty is to create a new skill. For this you can click the 'Create new skill' button that will open a new page. This page will ask you for a skill name and a skill template (pick, place, insert, ...). After this step, you will enter a new window which is used to configure the skill.

Each skill consists of multiple primitive. As an example, a 'Pick skill' will consist of 4 primitives:

- Movearm (cartesian motion to an approach location)
- Movearm (cartesian motion to a grasp location)
- Grasp (close gripper)
- Movearm (cartesian motion to a release location)

For each of those primitive, you will need to parameterize it (teach pose, select, velocity, ...)

In the following picture, you can see the configuration page. On the left side, you can see the primitive included in the skill. If you click on a primitive, the screen will now show all the required parameters.

Let's see how you can parameretize a movearm (PTP). The first line show the type of primitive. Below this, you can find the id of the position which will be stored in the DB. If you have created a new skill, this should be empty. The first step is then to generate one by teaching a point. To do this, you need to click 'Teach Pose'. This will trigger a kinestesic teaching request on the robot side. You can then click and hold the white button on the robot flange in order to move it at the right approach pose. When the button is release, the robot pose is sent back to the framework and uploaded in the database. If you refresh the web page, the id should be non-empty.

In addition, the line below (which shows the x, y, z, ... values should also be filled with the parameters). Below you can adjust the velocity by sliding the bar.

Once the id of the pose is avaiable, you can fully adapt this pose by different means:

- You can reteach the pose with the same procedure as before (Teach pose)
- You can manually enter the X,Y,Z,... values
- You can click on the Jog+/Jog- button. This is a quick method for jogging the robot flange up and down. This is relevent mainly when teaching approach/extract pose from the grasp pose.

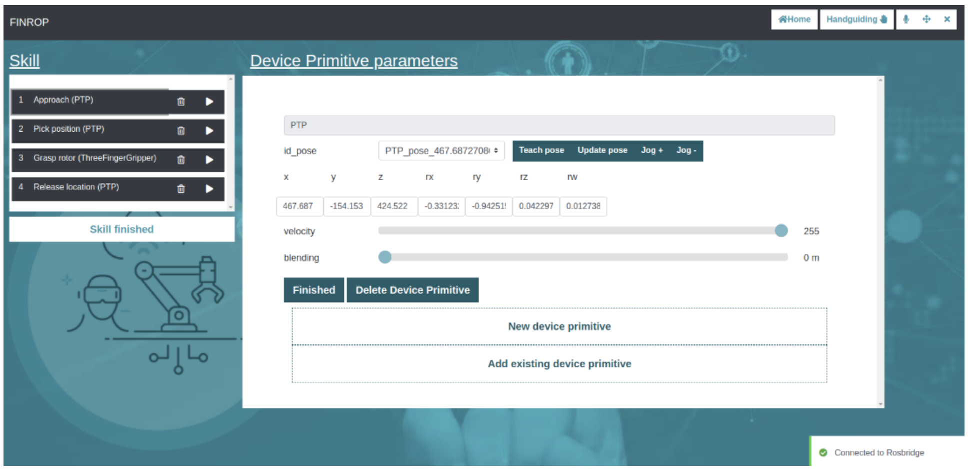
Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

- A final option is to reuse an existing point from the list (e.g. same extract and approach pose). Be careful that if you modify this pose and the point is used in another primitive, the changes will also affect this later

After having made some modifications, you can execute the primitive only in order to validate its parametrization. For this, you can click on the small play button corresponding to the right primitive (upper left corner). Once the parameterization is over, you can press the finished button and program the next primitive.

The other skill parameterization is staight forward except for the Trajectory skill which requires to teach multiple point at once. If you select the 'Trajectory skill' template, then at the start of the configuration page, the robot will enter in hand guiding mode. The procedure is similar to a regular 1 pose teach however, when you release the white button flange, a point will be added to the trajectory but the robot will remains in teaching mode. As such you can teach as many points as needed. If you finished your trajectory, you need to keep the white button holded and now pressed the green button followed by a release of the white button. This will exit the hand guiding mode and save the trajectory as a set of joint positions. If you refresh the page, all those points will be seen as individual primitives. You can modify each point and delete those if needed.

Another important features is that ability to manually add primitive to a skill. For this, click the 'New device primitive' button and select the type of primitive you want. This will be added by default at the end of the existing skill. However, you can click and drag that primitive at the appropriate location.



PTP

id_pose PTP_pose_467.6872708

x	y	z	rx	ry	rz	rw
467.687	-154.153	424.522	-0.33123	-0.94251	0.042297	0.012738

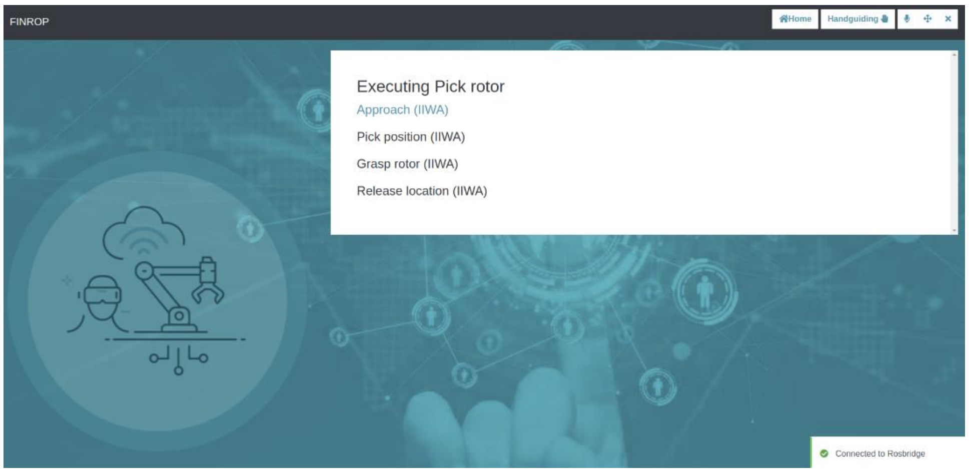
velocity 255

blending 0 m

When all the primitive in the skill are taught, you can click the 'Skill finished' button. You will enter back the main Skill list page. From there you can execute the skill you've just taught.

When the skill is being executed, you can keep track of which primitive is being executed.

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.



Executing Pick rotor

Approach (IIWA)

Pick position (IIWA)

Grasp rotor (IIWA)

Release location (IIWA)

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

4 Device

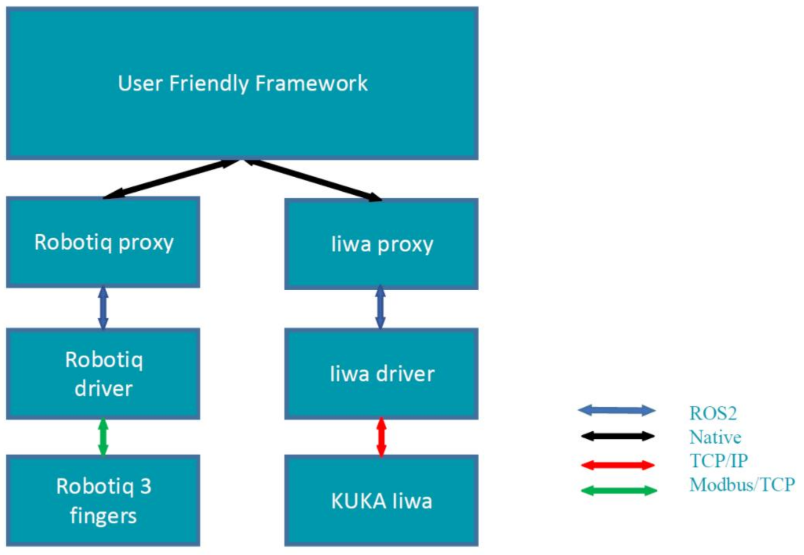
In this project, two devices are considered:

- A collaborative robot: KUKA Iiwa
- A collaborative gripper: 3 fingers Robotiq gripper

The below figure shows a simplified overview of the different devices and how they are communicating with the user-friendly framework.

The communication between the devices and the framework is split into different steps:

- A proxy layer is native to the framework and is responsible for sending ROS2 request to the driver layer
- A driver layer is listening for ROS2 request from the proxy layer and is responsible for translating the request to a communication protocol understandable by the devices
- A device layer that listens to the driver layer. In this project, the Robotiq 3f gripper is controller via Modbus RTU and the KUKA Iiwa via TCP/IP



4.1 Devices overview

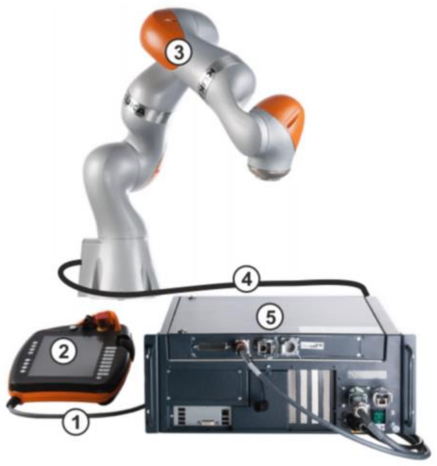
This section will focus on how the devices are programmed in order to receive commands from the above layers and interpreter those. This framework is a skill-based framework where a robotic task is composed of skills which is itself composed of device primitives. This latter is the command that is sent from the proxy all the way to the physical device. The device is therefore programmed in order to understand the corresponding device primitives and extract the parameters of the motion in order to perform it.

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

4.1.1.1 KUKA Iiwa

The following picture shows the hardware that is necessary for running an application on a KUKA Iiwa:

1. Connecting cable to the smartPAD
2. KUKA smartPAD control panel
3. Manipulator
4. Connecting cable to KUKA Sunrise Cabinet robot controller
5. KUKA Sunrise Cabinet robot controller



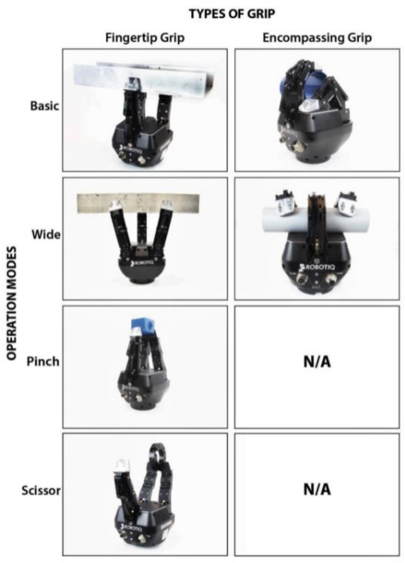
The programming and the configuration of the KUKA Iiwa are done using Sunrise Workbench which is a Java-based software developed by KUKA. This software runs on a remote computer. After properly configured and programmed the robot, the project can be synchronized with the KUKA Sunrise Cabinet.

3.1.2 Robotiq 3 fingers gripper

The Robotiq 3 fingers gripper is used as an independent agent in the work cell. Modbus RTU protocol is used in order to send a motion request or to get the status of the gripper.

As this gripper has 3 independent fingers, various grasping methods are possible. The following picture shows the available operation modes. In addition, this selected gripper is a collaborative gripper. As a result, the force of the motion can be set and this gripper has to ability to detect whether an object was correctly grasped. Besides the force and distance between the fingertips, the velocity of the motion is also a parameter.

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.



4.2 Hardware connection overview

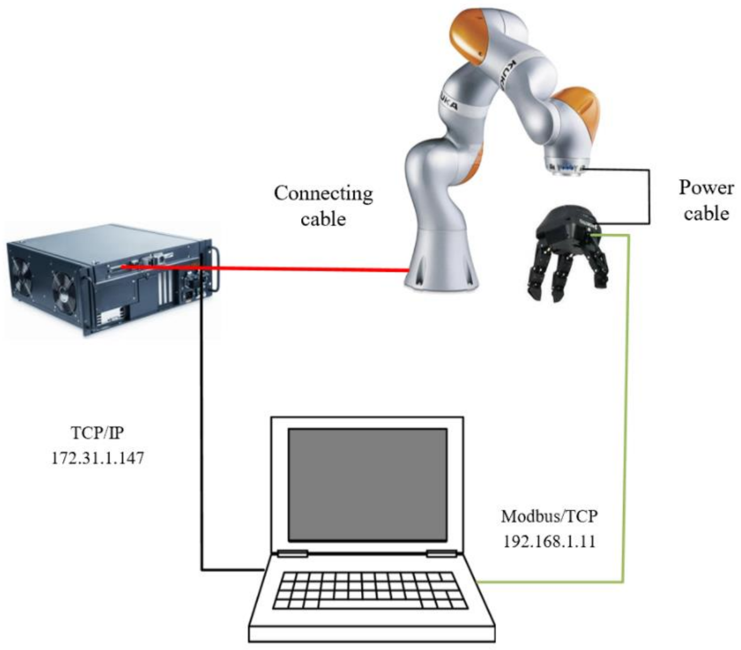
This section focuses on how to connect all the devices together with a correct wiring. This project is using several computers that need to be properly configured in other to communicate with each other. The main computer that is running the framework, as well as the HMI and database, needs to communicate both through TCP/IP (KUKA Iiwa) and Modbus RTU (Robotiq). Those two protocols are ethernet-based protocols. This means that the main computer needs to be physically linked to the two devices. The device computers (Robotiq, KUKA Cabinet) have their own IP address that can be configured. The main computer must be able to ping both IPs simultaneously.

In this project the IP address of the device computers are defined as follow:

- Robotiq gripper: 192.168.1.11
- KUKA Cabinet: 172.31.1.147

The following scheme shows how to connect everything together

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.



How can see that 2 ethernet based protocol are used requiring to connect 2 ethernet cables to the PC. A solution for this is to use an RJ45 to USB adaptor. A proper alternative is to add a router in between such that you can properly configure the ability to ping both IP class from your computer.

4.3 Overview of devices software

4.3.1.1 KUKA IIWA

The code running on the KUKA IIWA comes from this open source git repository

https://github.com/Modi1987/KST-Kuka-Sunrise-Toolbox/

The code running of the KUKA Cabinet is driven by two main codes:

- **BackgroundTask.java:** Is mainly responsible for the socket connection. This code is running in BackGround and listen to requests from the client (Framework request). When a request is received, a variable containing the serial data from the TCP/IP is initialized.
- **MatlabToolBoxServer.java:** This code is the main code in the project. It consists in a large loop that is constantly listening to updates of the variable containing the socket data. Depending on the variable content, a condition is the code will be triggered and be responsible for send the correct request to the robot (ex: MovePTP). The second role is to extract the

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.

parameters of the motion. Depending on the type of device primitives to perform the variable is trimmed in order to extract the parameters that defined the motion (position, velocity, force, ...)

4.3.1.2 Run the application (Robot side)

To run the application on the robot side, you only need to select the right code to execute from the Teach pendant. In this application the main application code to launch is the MatlabToolBoxServer.java file. Once selected on the Teach pendant, you need to press the play button.

The framework code needs to be started within a minute after the launch of the robot code. If this is not the case, the robot code will stop, and you will need to re-press play again.

4.3.1.3 Run the application (Gripper side)

The gripper does not need to be launch. The Modbus RTU command are received by the inner gripper controller that performed the command. Make sure you can correctly ping the gripper before launch the main framework application.

In case of any question please contact trinity@flandersmake.be

Trinity project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 825196.