FME TUL

PROJECT FISHER

USER MANUAL

Author: **Bc. Václav Hanzlík**

Supervisor: Ing. Miroslav Vavroušek, Ph.D.

Liberec 2023

Table of Contents

List of Figures

List of Tables

1 Introduction

Project Fisher is a Unity based application that enables you to control a digital version of Training Factory Industry 4.0 from Fischertechnik. It's used to simulate a real factory but on a small scale. The model can be controlled either locally or commands can be sent over network via the UDP protocol.

With this manual you will learn how to use the application and how to control the individual parts of the factory with commands.

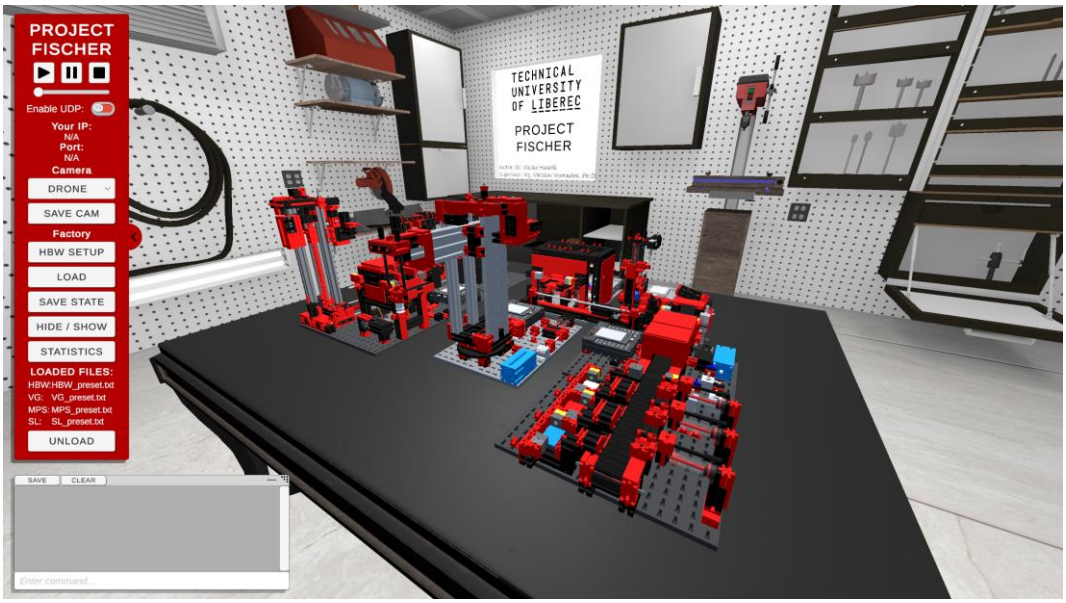

Fig. 1 - Main application window

1.1 Website – Download, examples

To download Project Fischer and to learn more about the application such as program examples for Python, C#, Matlab, Java or Unity go to this link:

pokrok.ksa.tul.cz/ProjectFischer

2 User interface

2.1 Side panel

Side panel is the main way for users to interact with the application and digital model. It consists of play controls, UDP management, camera controls and factory management. The panel can be closed and reopened by clicking the arrow. This chapter describes what all the different buttons do and how you can interact with the corresponding opened panels.

Fig. 2 - Side control panel

PLAY	Starts executing programs that are loaded in local memory or resumes paused application.
PAUSE	Pauses the application (Programs and statistics logging)
STOP	Stops executing programs and resets the factory.
Fast-Forward slider	Speed up the game by up to 4x.
Enable UDP	Switch to enable receiving commands over UDP.
Your IP	IP address of your computer. Note: Can be your VPN.
Port	The application listens for UDP messages on this port.
Camera dropdown	Choose a camera from a dropdown menu.

Fig. 3 - Camera dropdown menu

- **DRONE** — Fly around the factory like a drone.
  Controls:
  Activate by holding down RMB (Right Mouse Button)
  Move with „W, A, S, D“
  Speed up by holding down „Shift“
  Zoom with mouse wheel
  Pan by holding down MMB (Middle Mouse Button)
- **VR** — Select VR when you have a headset connected
  When in VR press „H“ to change scale.
  REQUIRES SteamVR to be installed on your system!
- **ROTATE** — Camera that rotates around the factory.
- **PRESET** — Pre-made cameras placed at points of interest.
- **CUSTOM** — If you create a custom camera it will show up here.

SAVE CAM Saves current DRONE camera position. Created camera can be selected from camera dropdown menu.

HBW SETUP Fill warehouse with parts or empty crates.

Fig. 4 - Warehouse setup screen

- **DEFAULT** Fills warehouse with white parts on top, red parts in the middle, blue parts on the bottom (WWWRRRBBB)
- **EMPTY** Deletes all crates and parts in warehouse
- **CUSTOM** Open custom warehouse window and select desired layout by clicking on the icons.

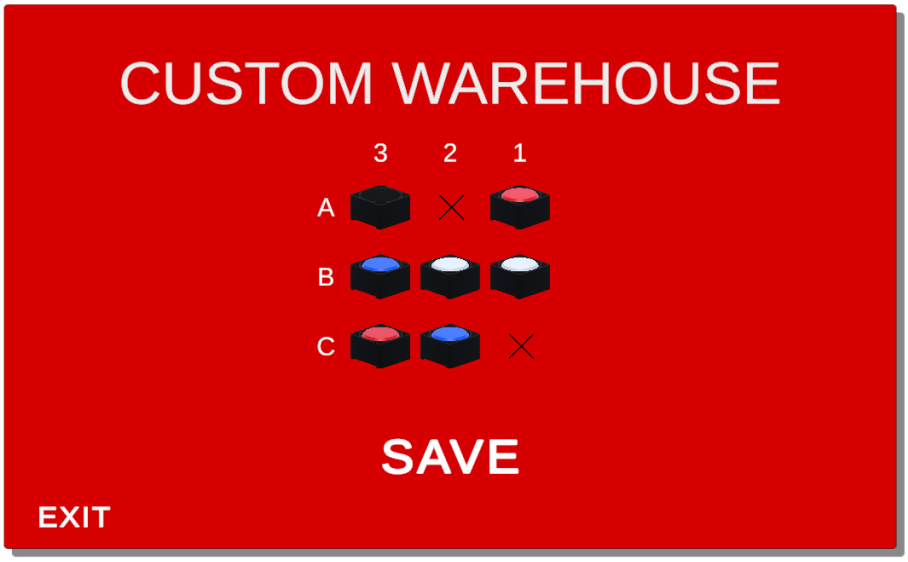

Fig. 5 - Custom warehouse screen

LOAD FILE Open Load window and select programs to load into memory. Click Browse to open file browser.

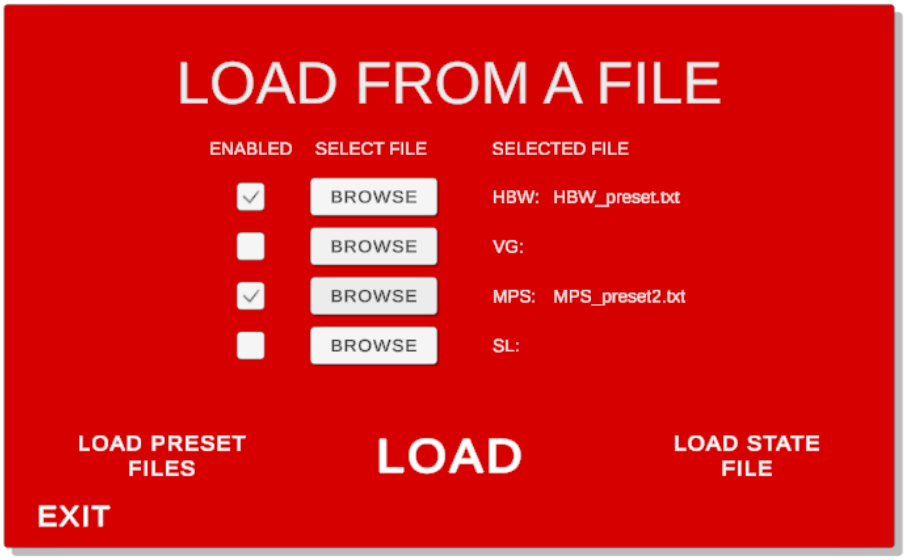

Fig. 6 - Load screen

LOAD PRESET FILES Load pre-made preset programs for all modules

LOAD STATE FILE Select file with saved state to load. (Loads instantly)

SAVE STATE Save current factory settings, cameras and positions to a file.

HIDE / SHOW Open hide/show window. Enable or disable factory modules by clicking on the icons.

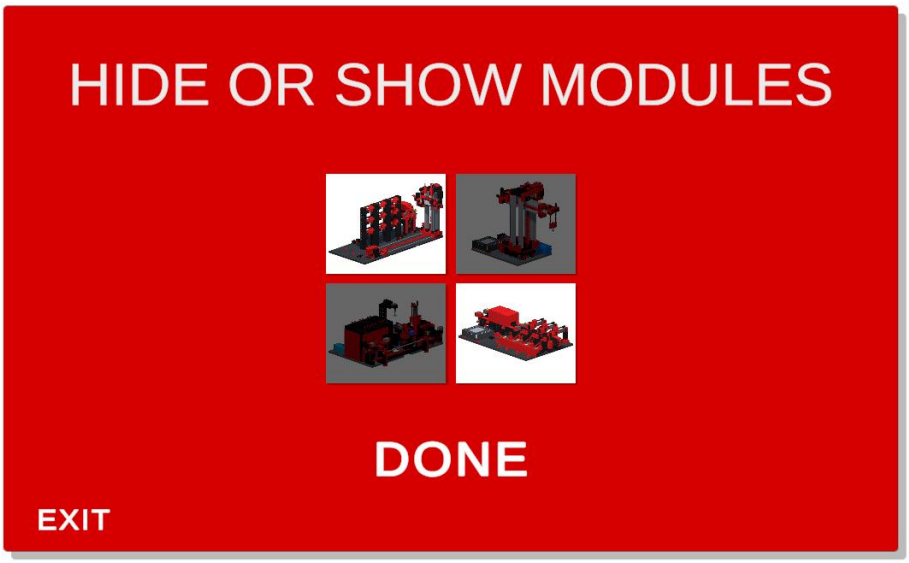

Fig. 7 - HIDE / SHOW screen

STATISTICS Open statistics window.

In the statistics window you can see all 13 motors that are in the factory. Under the engine name there is a pie chart that represents the ratio between time spent working (moving) and idle time (stopped). Next to the pie chart is a number representation of the time and percentage.

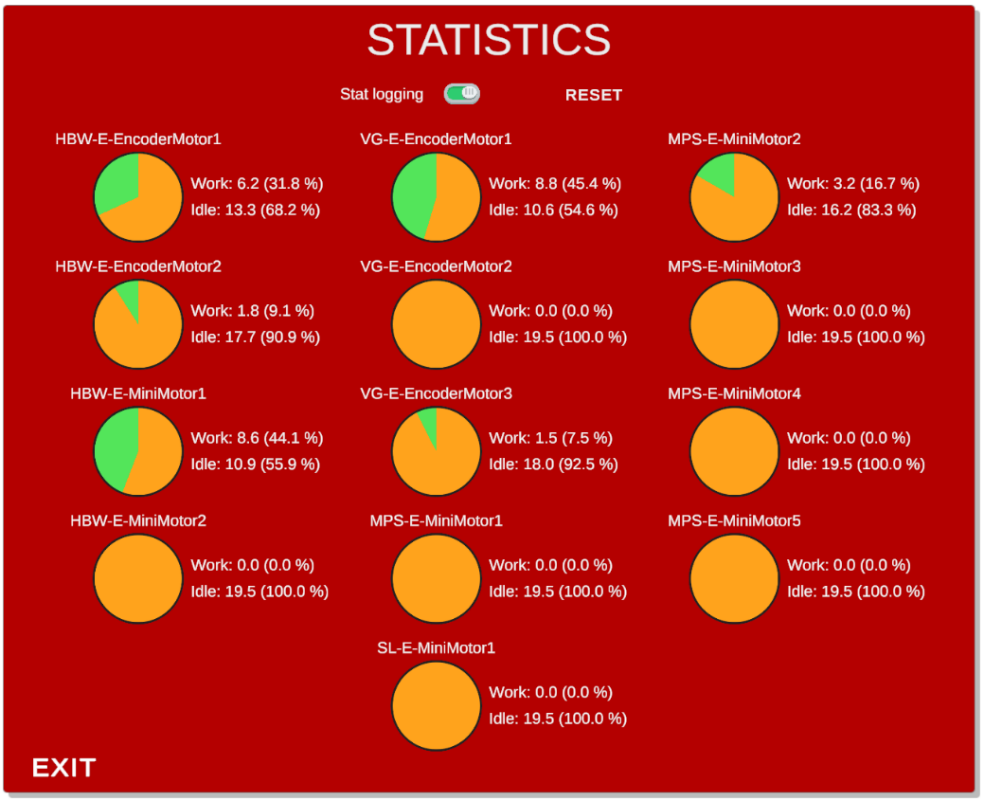

Fig. 8 - Statistics screen

UNLOAD Unload all programs currently in memory.

2.2 Console

Console is great for learning what has happened and is currently happening to the digital model. To process your command type it inside the input field and submit by pressing ENTER.

There are three types of messages you can encounter:

- [LOCAL] – Commands that were executed locally from a console or a file.
- [UDP] – Commands that were received via the UDP protocol.
- Other – Error messages, factory status etc.

The console also features a command history. To access this, click inside the input field and use arrow keys (UP and DOWN) to browse the previously used commands in your current session.

The window can be dragged by the top handle, minimized by clicking the horizontal line and resized by dragging the dotted triangle in top right corner.

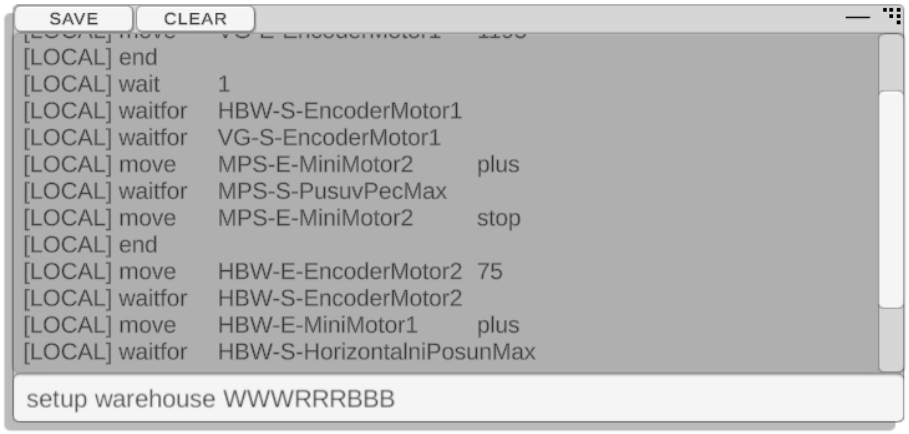

Fig. 9 - Console panel

SAVE	Save console output to a .txt file
CLEAR	Clear the console window (irreversible)

3 Motors and sensors overview

The factory features 4 modules, each with their own motors and sensors that can be controlled by a simple programming language.

In this chapter you can learn about the names of the motors and sensors. You can then use this knowledge to reference them in your program.

3.1 High Bay Warehouse (HBW)

3.1.1 Motors

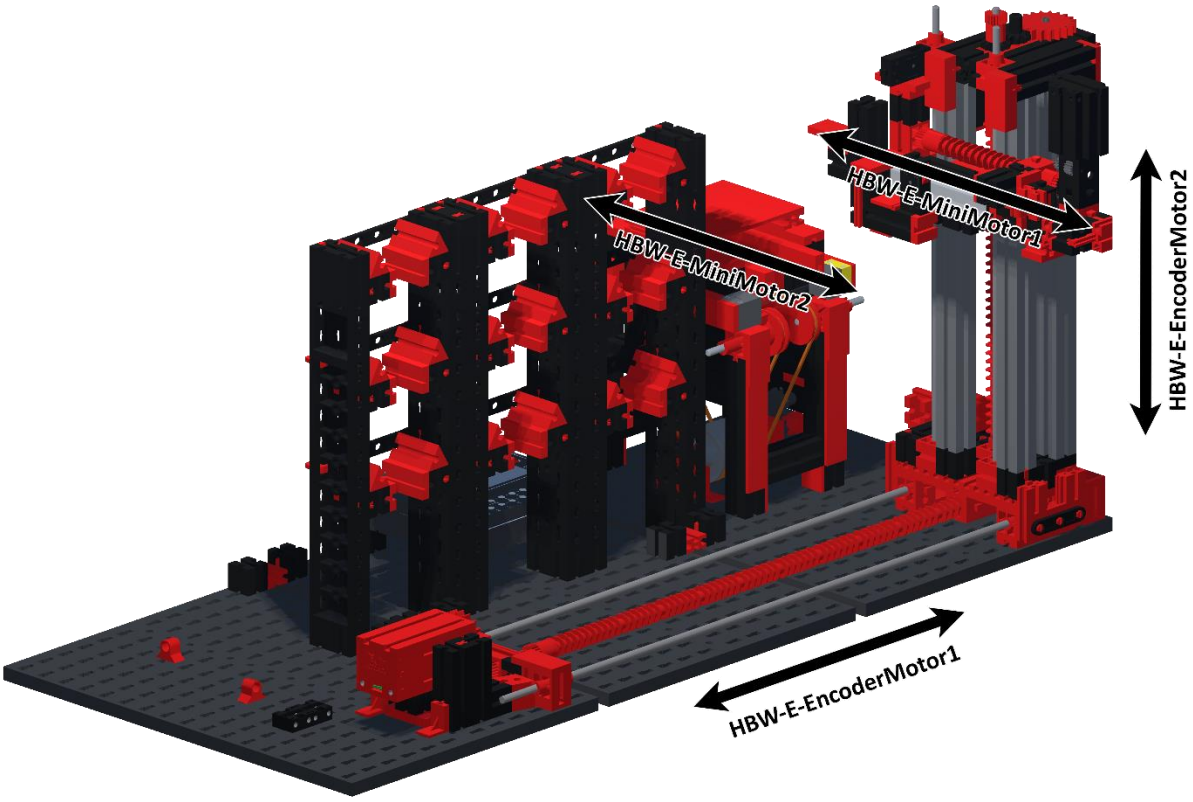

Fig. 10 - HBW motors

Table 1 - List of HBW motors

High Bay Warehouse			
Motor name	**Type**	**Min value**	**Max value**
HBW-E-EncoderMotor1	Encoder	0	2000
HBW-E-EncoderMotor2	Encoder	0	870
HBW-E-MiniMotor1	Mini Motor	-	-
HBW-E-MiniMotor2	Mini Motor	-	-

3.1.2 Sensors

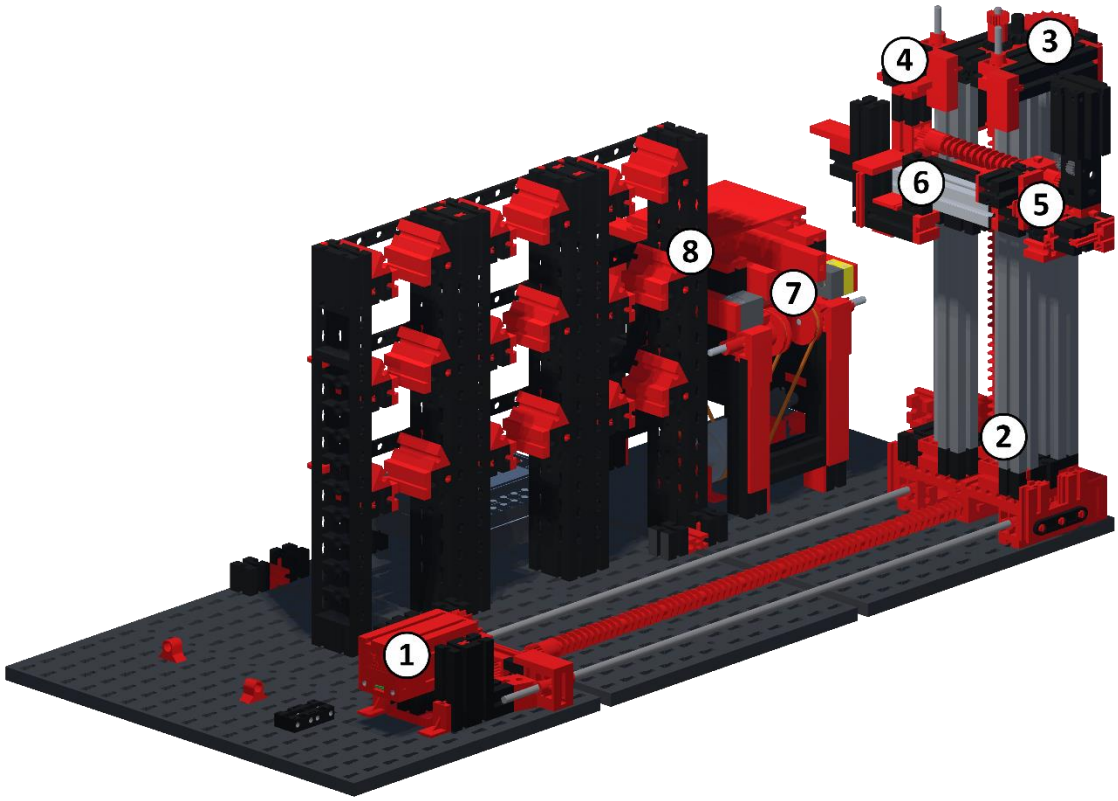

Fig. 11 - HBW sensors

Table 2 - List of HBW sensors

High Bay Warehouse		
Number	**Sensor name**	**Type**
1	HBW-S-EncoderMotor1	Encoder
2	HBW-S-HorizontalniPosunHlavni	End
3	HBW-S-EncoderMotor2	Encoder
4	HBW-S-VertikalniPosun	End
5	HBW-S-HorizontalniPosunMin	End
6	HBW-S-HorizontalniPosunMax	End
7	HBW-S-LightSensor1	Light
8	HBW-S-LightSensor2	Light

3.2 Vacuum Gripper (VG)

3.2.1 Motors

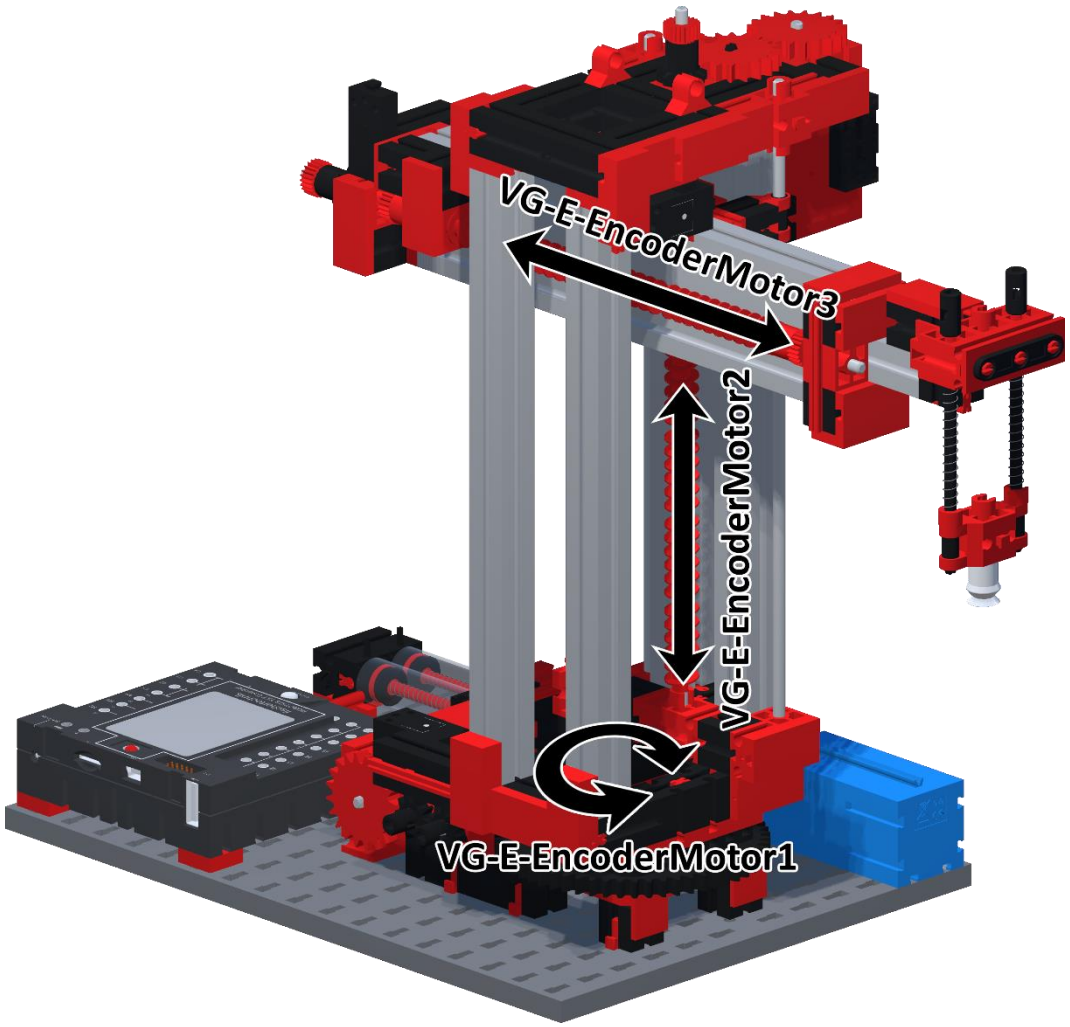

Fig. 12 - VG motors

Table 3 - List of VG motors

Vacuum Gripper			
Motor name	**Type**	**Min value**	**Max value**
VG-E-EncoderMotor1	Encoder	0	1580
VG-E-EncoderMotor2	Encoder	0	850
VG-E-EncoderMotor3	Encoder	0	940
VG-P-Valve1	Valve	0	1

The **VG-P-Valve1** is used to create a vacuum for picking up parts with the effector.

3.2.2 Sensors

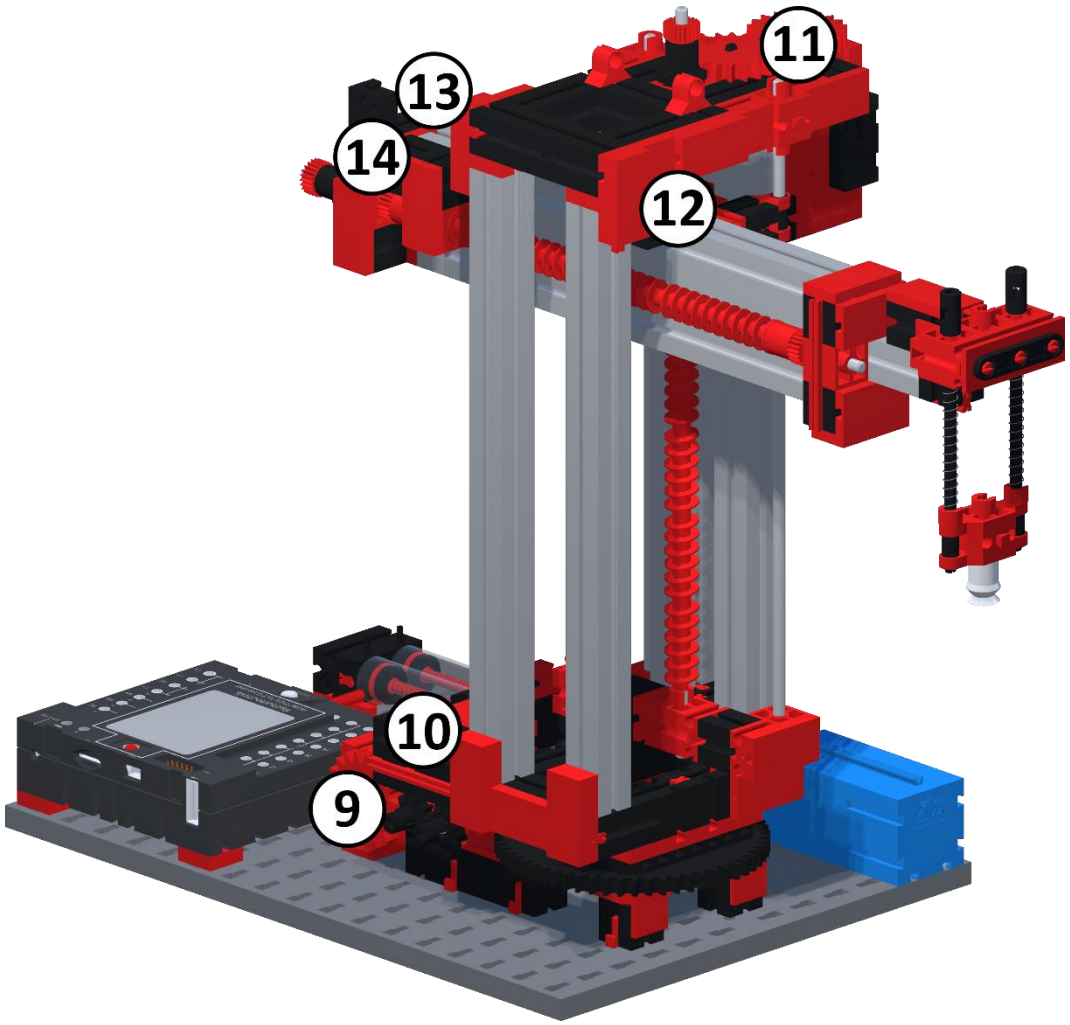

Fig. 13 - VG sensors

Table 4 - List of VG sensors

Vacuum Gripper		
Number	**Sensor name**	**Type**
9	VG-S-EncoderMotor1	Encoder
10	VG-S-HlavniRotace	End
11	VG-S-EncoderMotor2	Encoder
12	VG-S-VertikalniPosun	End
13	VG-S-EncoderMotor3	Encoder
14	VG-S-HorizontalniPosun	End

3.3 Multi-Processing Station (MPS)

MPS is the only module that has a programmable light: **MPS-L-OvenLight**. It is located inside the oven. The light can be controlled with the command „set“ and has two possible states: 1 (ON) / 0 (OFF)

3.3.1 Motors

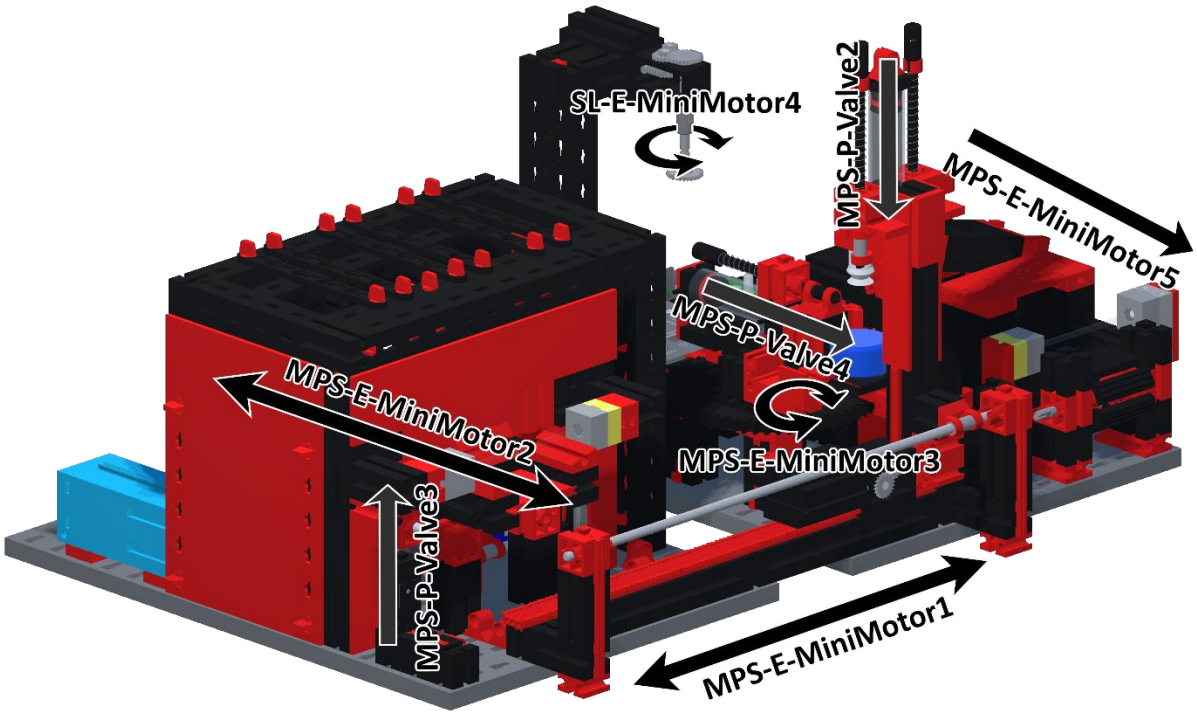

Fig. 14 - MPS motors

Table 5 - List of MPS motors

Multi-processing Station			
Motor name	**Type**	**Min value**	**Max value**
MPS-E-MiniMotor1	MiniMotor	-	-
MPS-E-MiniMotor2	MiniMotor	-	-
MPS-E-MiniMotor3	MiniMotor	-	-
MPS-E-MiniMotor4	MiniMotor	-	-
MPS-E-MiniMotor5	MiniMotor	-	-
MPS-P-Valve1	Valve	0	1
MPS-P-Valve2	Valve	0	1
MPS-P-Valve3	Valve	0	1
MPS-P-Valve4	Valve	0	1
MPS-L-OvenLight	Light	0	1

The **MPS-P-Valve1** is used to create a vacuum for picking up a part with the effector.

3.3.2 Sensors

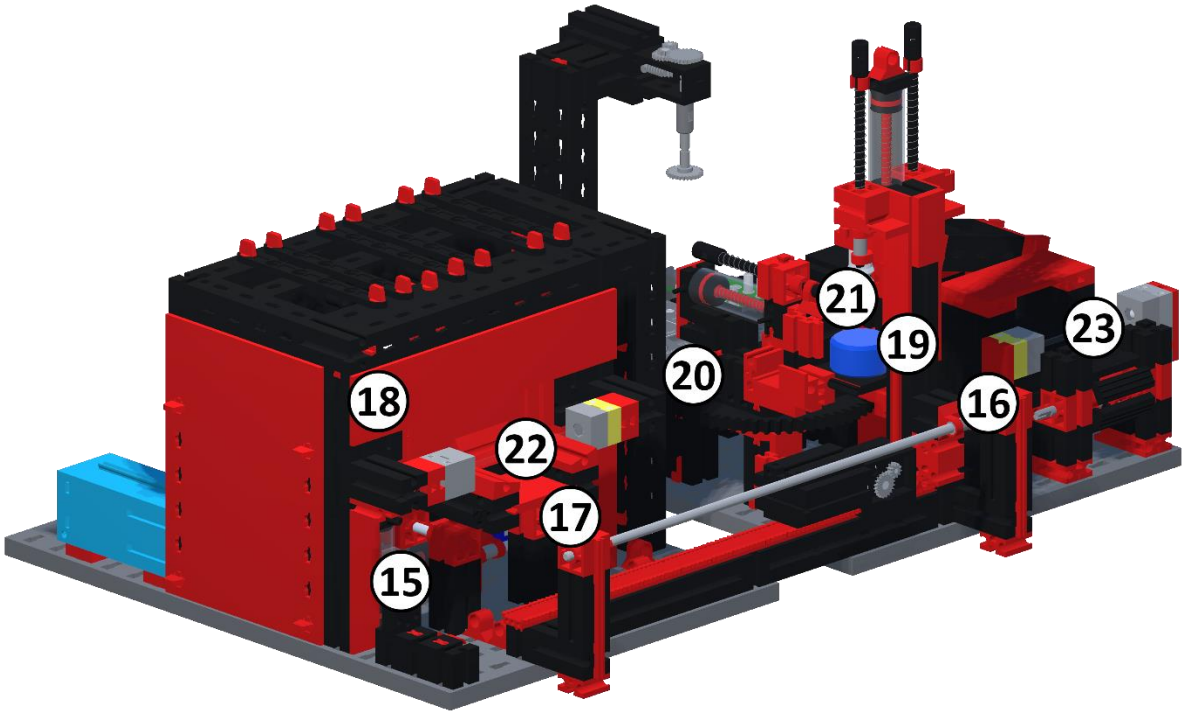

Fig. 15 - MPS sensors

Table 6 - List of MPS sensors

Multi-processing Station		
Number	**Sensor name**	**Type**
15	MPS-S-HorizontalniPosunMax	End
16	MPS-S-HorizontalniPosunMin	End
17	MPS-S-PusuvPecMin	End
18	MPS-S-PusuvPecMax	End
19	MPS-S-RotacniStulMin	End
20	MPS-S-RotacniStulMid	End
21	MPS-S-RotacniStulMax	End
22	MPS-S-LightSensor1	Light
23	MPS-S-LightSensor2	Light

3.4 Sorting Line (SL)

3.4.1 Motors

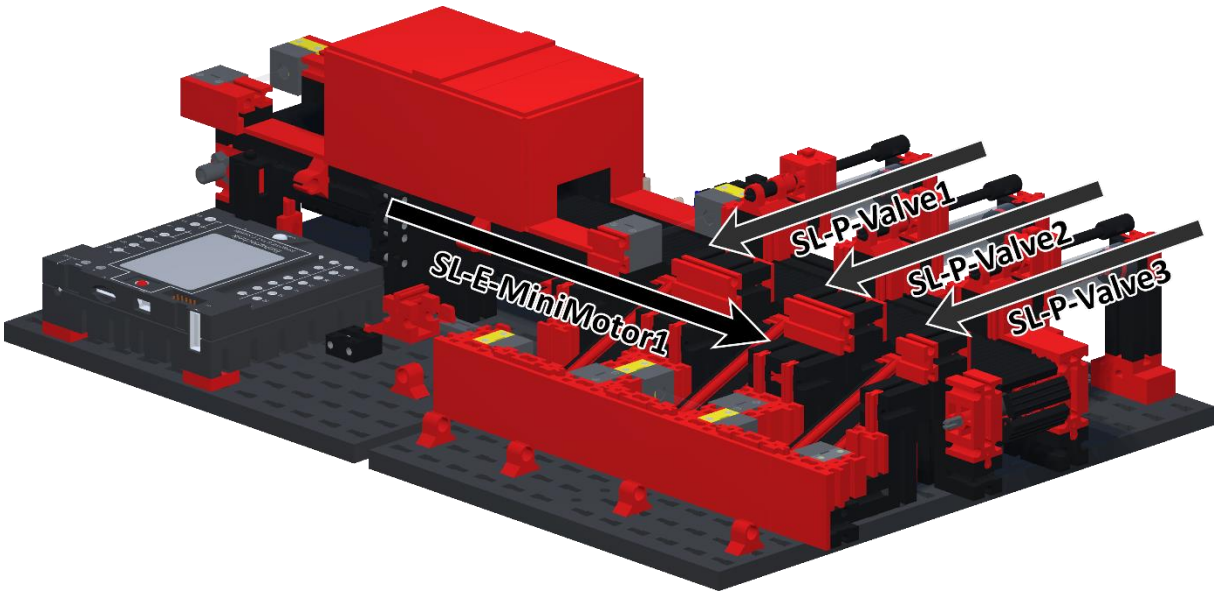

Fig. 16 - SL motors

Table 7 - List of SL motors

Sorting Line			
Motor name	**Type**	**Min value**	**Max value**
SL-E-MiniMotor1	MiniMotor	-	-
SL-P-Valve1	Valve	0	1
SL-P-Valve2	Valve	0	1
SL-P-Valve3	Valve	0	1

3.4.2 Sensors

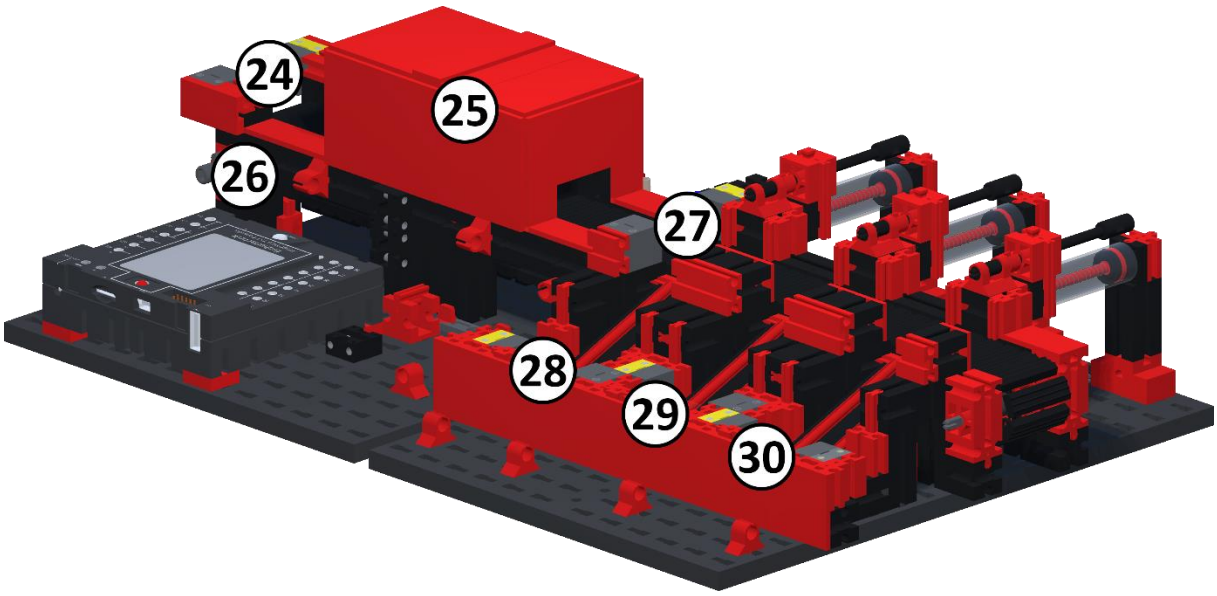

Fig. 17 - SL sensors

Table 8 - List of SL sensors

Sorting Line		
Number	**Sensor name**	**Type**
24	SL-S-LightSensor1	Light
25	SL-S-ColorSensor	Color
26	SL-S-PulseCounter	Pulse
27	SL-S-LightSensor2	Light
28	SL-S-LightSensor3	Light
29	SL-S-LightSensor4	Light
30	SL-S-LightSensor5	Light

4 Programming

Project Fischer uses a custom programming language designed to control the digital model. Note that commands and object names are not case sensitive, but do require correct spelling.

4.1 List of commands

Table 9 - List of commands

Function	Object	Parameter	Description
start			Starts executing programs in local memory
move	XYZ-E-EncoderMotorX	*no. pulses*	Moves motor smoothly to location
	XYZ-E-MiniMotorX	plus	Moves motor in plus direction
		minus	Moves motor in minus direction
		stop	Stops motor
set	XYZ-E-EncoderMotorX	int *pulses*	Moves motor instantly to location
	XYZ-P-ValveX	1	Opens valve (Extends piston)
		0	Closes valve (Retracts piston)
	MPS-L-OvenLight	1	Turns on light
		0	Turns off light
wait		*no. seconds*	Waits for X seconds (in-game)
waitfor	XYZ-S-SensorNameX		Waits for sensor activation
get	XYZ-S-SensorNameX		Returns sensor status
sync	XYZ-E-EncoderMotorX	*no. pulses*	Sync motor instantly to location during move to required value, only for UDP
repeat			Repeats the program from the top line
end			Stops program from reading next line
reset			Resets factory
setup	camera	X Y Z rotX rotY rotZ	Creates custom camera at position (X, Y, Z) and rotation (rotX, rotY, rotZ)
	warehouse	XYZXYZXYZ	Fills warehouse (E-Empty, C - Crate, W - White, R - Red, B - Blue)
	show	*module*	Shows module (HBW, VG, MPS, SL)
	hide	*module*	Hides module (HBW, VG, MPS, SL)
log	start		Starts logging statistics
	stop		Stops logging statistics
	reset		Resets statistics
	get	*XYZ-E-EncoderMotorX*	Returns statistics for motor

4.2 Creating a program

By combining the function, object and/or a parameter into a string **separated by either a space or a tab** you can control the digital model and the whole application. You can input each line of the code into console or you can create a text file with all the commands you want. The application will read each line from the top of the text file.

Encoder motors require you to input the desired value in *pulses*. That's because the motor uses a hall sensor to measure angle increments. It has a built-in gearbox with a ratio of 21.3:1. The encoder produces 3 pulses for one revolution of the motor. Therefore, 63.9 pulses are required for one rotation of the output shaft from the gearbox. Or you can use directional input (plus / minus)

Mini motors on the other hand only support directional input.

As an example, here is a part of a program built for the HBW module:

```
setup       warehouse                   WWWRRRBBB
move        HBW-E-EncoderMotor1         780
waitfor     HBW-S-EncoderMotor1
move        HBW-E-EncoderMotor2         75
waitfor     HBW-S-EncoderMotor2
move        HBW-E-MiniMotor1 plus
waitfor     HBW-S-HorizontalniPosunMax
move        HBW-E-MiniMotor1 stop
move        HBW-E-EncoderMotor2         5
waitfor     HBW-S-EncoderMotor2
move        HBW-E-MiniMotor1 minus
waitfor     HBW-S-HorizontalniPosunMin
move        HBW-E-MiniMotor1 stop
move        HBW-E-EncoderMotor1         20
waitfor     HBW-S-EncoderMotor1
move        HBW-E-MiniMotor1 plus
waitfor     HBW-S-HorizontalniPosunMax
move        HBW-E-MiniMotor1 stop
move        HBW-E-EncoderMotor2         920
waitfor     HBW-S-LightSensor1
waitfor     HBW-S-EncoderMotor2
move        HBW-E-MiniMotor2 plus
waitfor     HBW-S-LightSensor2
move        HBW-E-MiniMotor2 stop
```

You can find all the default programs (loaded with the LOAD PRESET FILES button in the LOAD panel) in the application root folder:

ProjectFischer\Project_Fischer_Data\StreamingAssets

- ↳ HBW_preset.txt
- ↳ VG_preset.txt
- ↳ MPS_preset.txt
- ↳ SL_preset.txt

5 UDP communication

The application can act as a UDP server and can receive commands from clients on the same network or over the internet when properly configured.

To send commands from a client use the **IP address of the computer** that runs Project Fischer and the default **port 8051**.
Python script example for sending a command:

```
serverAddressPort   = ("192.168.0.101", 8051)

def Send(data:str)->str:
    UDPClientSocket.sendto(str.encode(data), serverAddressPort)

Send("move  HBW-E-EncoderMotor1   780")
```

Examples in more languages are available on website mentioned in chapter 1.1.

6 Error messages

Here are some of the error messages you might encounter while executing commands or programs along with a possible solution.

Table 10 - List of error messages

Error code	Solution
No programs loaded	Check if there are programs loaded in the left panel
Unknown command	See the list of commands for supported phrases
Unknown engine	Check spelling of the engine name
Unknown sensor	Check spelling of the sensor name
Unknown module	Check spelling of the module abbreviation
Invalid value	Value is outside of the limits