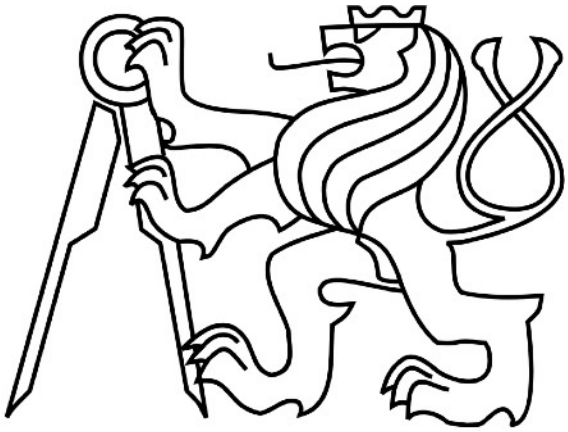
Czech Technical University
in Prague



Faculty of Mechanical Engineering

Department of Mechanics, Biomechanics and Mechatronics

Automated control of adjustable mechanical elements of counting machines using LEGO Mindstorms components

Master's Thesis

Sára Strakošová

Prague
2022

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Strakošová** Jméno: **Sára** Osobní číslo: **473547**

Fakulta/ústav: **Fakulta strojní**

Zadávající katedra/ústav: **Ústav mechaniky, biomechaniky a mechatroniky**

Studijní program: **Aplikované vědy ve strojním inženýrství**

Specializace: **Mechatronika**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Automatizované ovládání nastavitelných mechanických elementů vibračních bubnů využívající komponenty LEGO Mindstorms

Název diplomové práce anglicky:

Automated control of adjustable mechanical elements of counting machines using LEGO Mindstorms components

Pokyny pro vypracování:

1. Seznamte se s koncepcí a řídícím systémem vibračních bubnů na LEGO Pre-Pack balicích linkách.
2. Navrhněte a zrealizujete úpravu stávajícího řešení horizontálních a vertikálních nastavovacích prvků (tzv. škrabek) tak, aby se místo aktuálně používaných prvků použily komponenty LEGO Mindstorms včetně servomotorů a řídicí jednotky.
3. Vytvořte řídicí aplikaci, která zajistí správné ovládání škrabek.
4. Seznamte se s průmyslovým komunikačním protokolem OPC UA.
5 .Navrhněte a implementujte řešení postavené na platformě Raspberry Pi a programovacím jazyce Python pro propojení LEGO Mindstorms jednotky s existujícím nadřazeným řídicím systémem komunikujícím pomocí protokolu OPC UA .
6. Vytvořte grafické rozhraní pro ovládání.
7. Otestujte vytvořený systém.

Seznam doporučené literatury:

[1] Mahnke, Wolfgang, Stefan-Helmut Leitner, and Matthias Damm. OPC unified architecture. Springer Science & Business Media, 2009.
[2] Pilgrim, Mark, and Simon Willison. Dive Into Python 3. Vol. 2. Apress, 2009.
[3] Informační zdroje dostupné na webu organizace OPC Foundation: https://opcfoundation.org/resources

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Petr Beneš, Ph.D. odbor mechaniky a mechatroniky FS

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **22.04.2022** Termín odevzdání diplomové práce: **15.08.2022**

Platnost zadání diplomové práce: _____

Ing. Petr Beneš, Ph.D. podpis vedoucí(ho) práce	prof. Ing. Michael Valášek, DrSc. podpis vedoucí(ho) ústavu/katedry	doc. Ing. Miroslav Španiel, CSc. podpis děkana(ky)

Declaration

I declare that this paper is my original authorial work, which I have worked out independently. All sources, references, and literature used or excerpted during elaboration of this work are properly listed.

Sára Strakošová

Acknowledgments

I would like to thank my supervisor Ing. Petr Beneš, Ph.D. and my consultant doc. Ing. Petr Kadera, Ph.D. for their guidance during work on my thesis. I would also like to thank Martin Jílek, Ph.D. for his help during practical parts of my work. And many thanks to my family friends for their support and help.

Supervisor: Ing. Petr Beneš, Ph.D.

Consultant: doc. Ing. Petr Kadera, Ph.D.

Abstract

This diploma thesis consists of two major parts, namely stepper motor replacement and scraper control system. Both main parts are mostly practical and are supplemented by a research part with purpose of acquainting the reader with the LEGO pre-pack line concept and the OPC UA communication standard. The goal of the practical part is to replace the stepper motor with a LEGO motor and create a new control system for scrapers' adjustment that communicates with a superior control system using the OPC UA technology.

Keywords

OPC UA, LEGO pre-pack line, Counting machine, servomotor, stepper motor

Table of contents

1 Introduction

Until recently, operation of counting machines relied mostly on workers. Because of a wide range of LEGO elements and due to the fact that every element needs different counting machine setting, it takes more than half a year for workers to learn how to operate a counting machine correctly and efficiently. In cooperation with CIIRC CTU, the counting machines have been modified and partially automated over the course of several projects. The process of scrapers' adjustment has been semi-automated with stepper motors. The goal of this thesis is to replace the stepper motors with LEGO motors which provide position feedback. This replacement would reduce not only the costs, but the human labour needed for operating the machines too.

The first part of this thesis deals with the issue of the stepper motor replacement in a mechanical way. The second part of the thesis deals with the issue of creating the control system that would connect all the parts that are needed for scrapers' adjusting, built on Raspberry Pi platform using Python programming language. The thesis also demonstrates the use of the OPC UA communications standard in unusual applications.

2 LEGO pre-pack line

LEGO pre-pack line packs LEGO elements in separate plastic bags, dividing elements into smaller amounts. Separated bags are then packed into LEGO sets in the Final pack department. The pre-pack method, introduced in the late 1970s, reduces the number of sets that contain an incorrect amount of LEGO elements. LEGO pre-pack line consists of three main parts: the counting machines, the scales and the bag makers. In recent years, the most attention was paid to the improvement of these three components' behavior and to the development of optimization methods for higher reliability and production quality. [1] [2]

2.1 Origin of the pre-pack line

Due to the increased pace of production in the 1970s, a high number of LEGO sets contained an incorrect amount of LEGO bricks, either too many or too few. The main reason for these high error rates was the fact that the majority of the packing process was done manually. The solution to this problem was the introduction of the pre-pack concept. LEGO elements were no longer packed all together by mixing small and large pieces, instead they were packed in smaller quantities into separate plastic bags. This packing method offered better control over the amount of packed elements and reduced manual work, thus ensuring a lower error rate. [1]

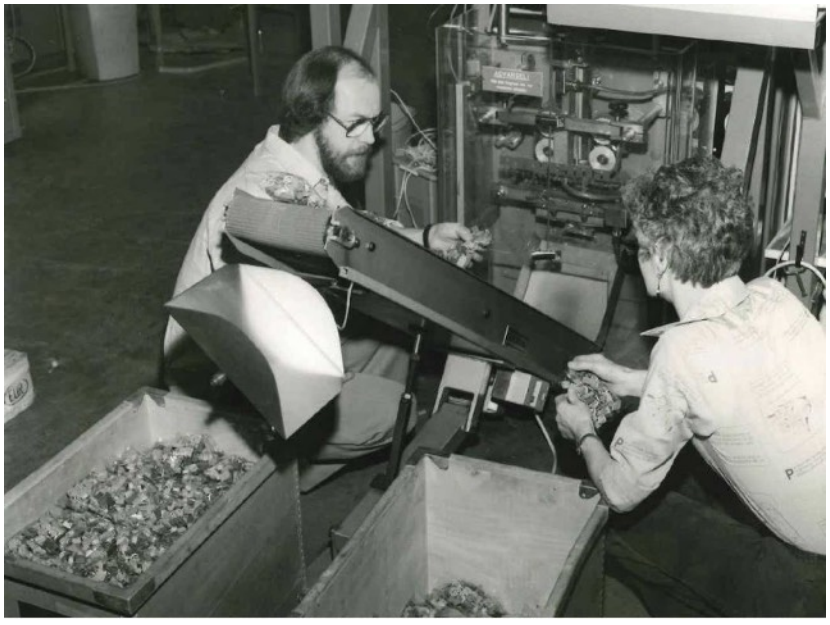


Figure 2.1: Early pre-pack line 1980 [1]

2.2 Counting machine

A key device of the LEGO pre-pack line is a counting machine. A counting machine separates LEGO elements using vibrations so that they can be counted and divided into the pre-pack packages in the right amount. LEGO elements that are to be sorted often change, so a counting machine must be easily adjustable. Two types of LEGO pre-pack lines are distinguished: the “single” and the “double”. On a “double” pre-pack line, there are 36 counting machines, on a “single” pre-pack line, there are 18 counting machines.

Figure 2.2: LEGO pre-pack line [3]

2.2.1 Construction

A counting machine consists of a spiral and a linear path through which LEGO elements are moved onto the belt conveyor. The movement of LEGO elements is secured by vibration actuators, one for each path. Two adjustable scrapers ensure the correct distribution of LEGO elements on the spiral path and the optical sensor controls the level of the LEGO elements that are supplemented to the middle of the spiral path. Touch display enables the vibration setting and scraper adjustment. All of these components are placed on a metal box with a control system inside of it. The main focus of this diploma thesis is the scrapers that are controlled by pneumatic brakes, air-flow system and motors.

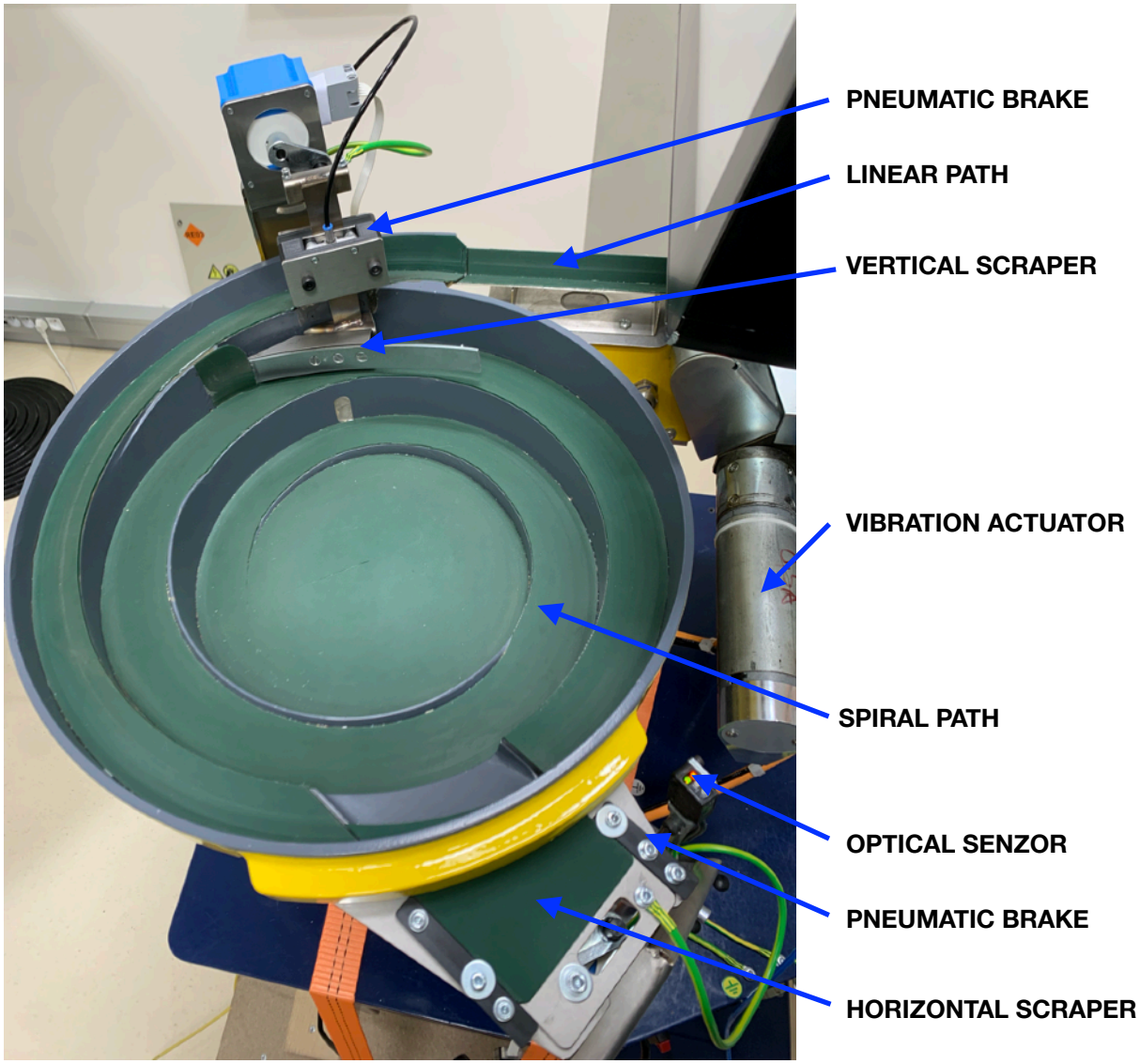


Figure 2.3: Counting machine - top view

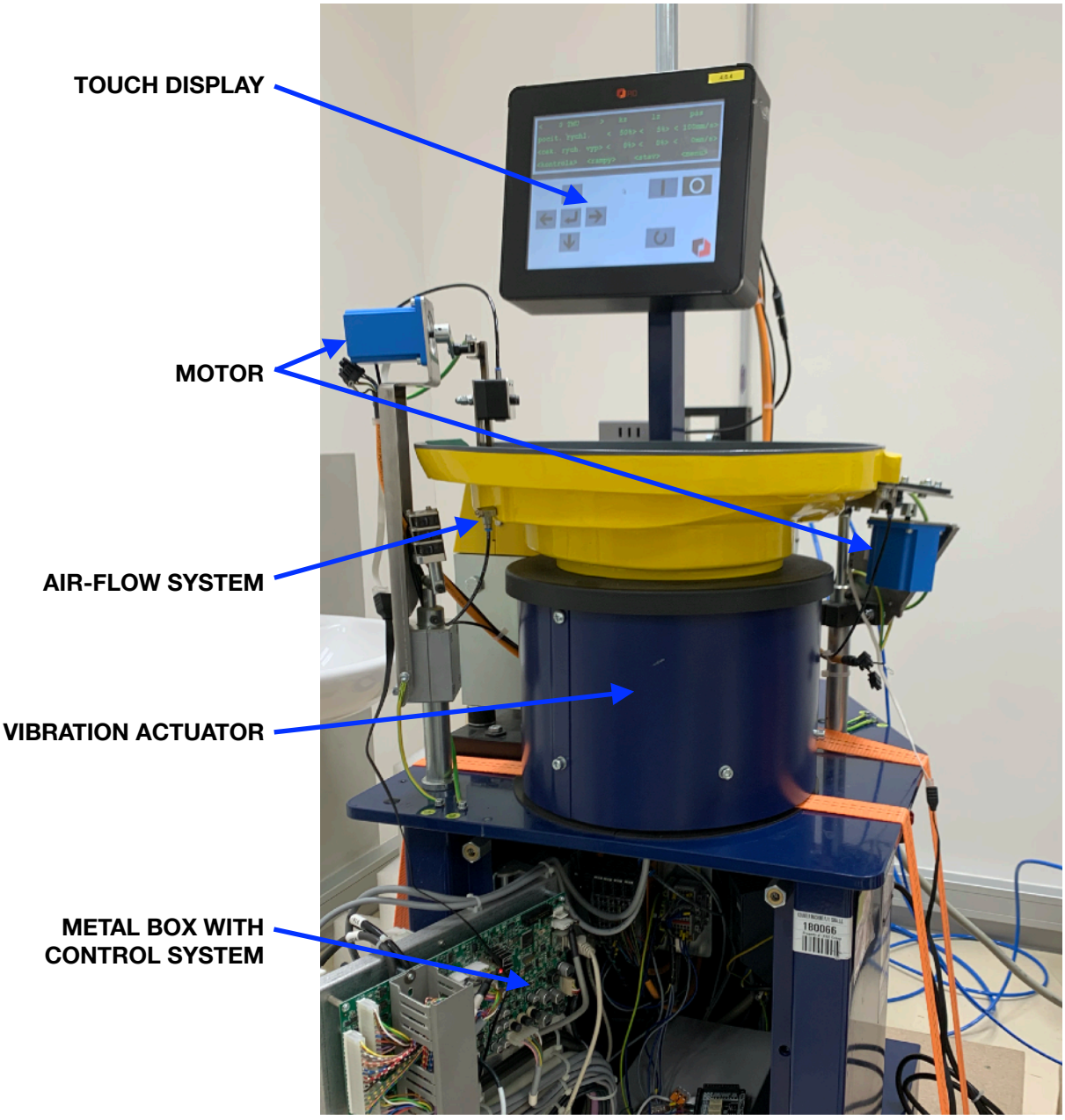


Figure 2.4: Counting machine - front view

Scrapers are adjustable mechanical elements distributed alongside the spiral path of a counting machine. Each counting machine has two scrapers. The first scraper moves vertically and is called the vertical scraper (figure 2.7). The second scraper moves horizontally, this scraper is called the horizontal scraper (figure 2.8). Both scrapers have the purpose of creating an obstacle on the spiral path. The vertical scraper prevents two or more LEGO elements from passing on top of each other, while the horizontal scraper prevents two or more LEGO elements from passing next to each other. Each scraper is connected to a motor by a crank that converts the rotational movement of a motor to the sliding movement of a scraper.

There is no risk of a LEGO element getting stuck when adjusting the horizontal scraper, but when adjusting the vertical scraper, the risk is high. If this complication occurs, it can be resolved by the air-flow system. When the air-flow system is engaged, the air stream blows in the vertical scraper area and blows away any stuck LEGO elements back to the center of the spiral path.

As shown in figures 2.5 and 2.6, when the scrapers are not being adjusted, the pin of the crank does not touch the scraper and is positioned in the middle of the slit. This isolates the motor from the vibrations. For the purpose of keeping the scrapers in their set position, there are pneumatic brakes that are always engaged unless the position is reset. Each scraper has one pneumatic brake.

Figure 2.5: Crank and slit - vertical scraper

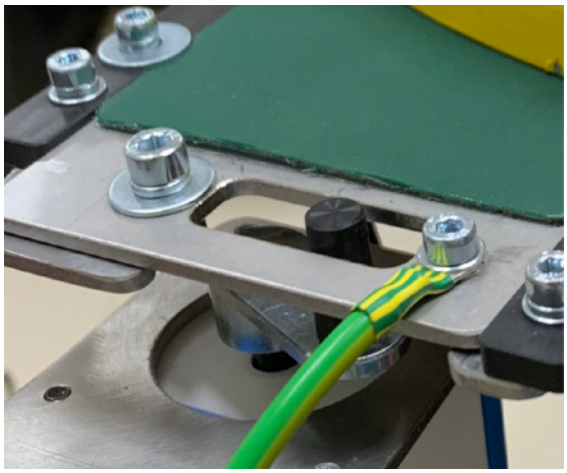


Figure 2.6: Crank and slit - horizontal scraper

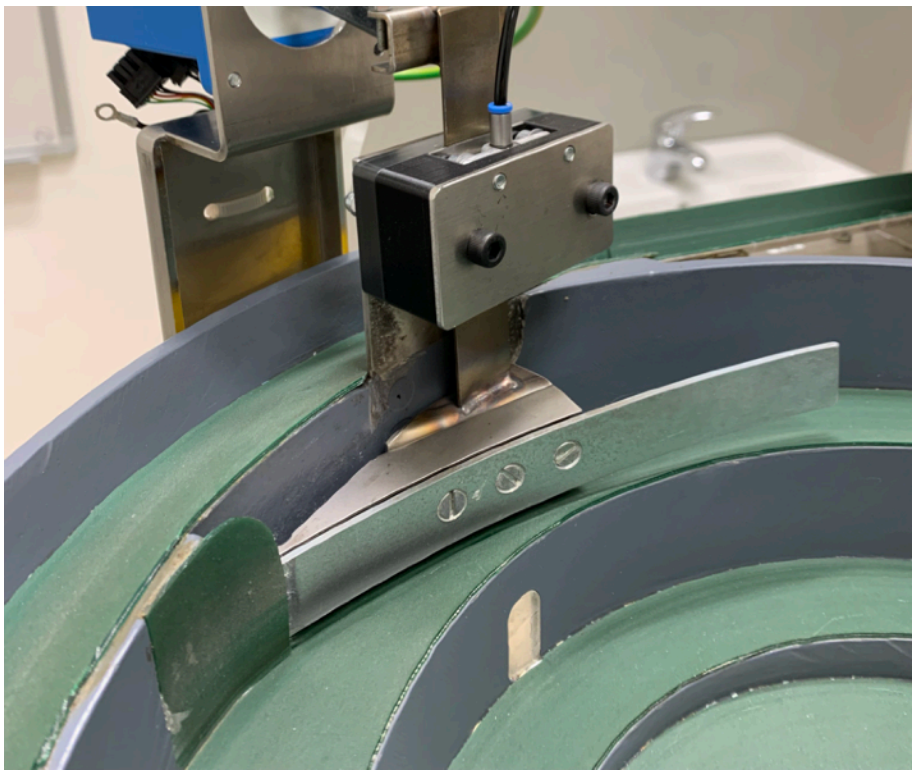


Figure 2.7: Vertical scraper

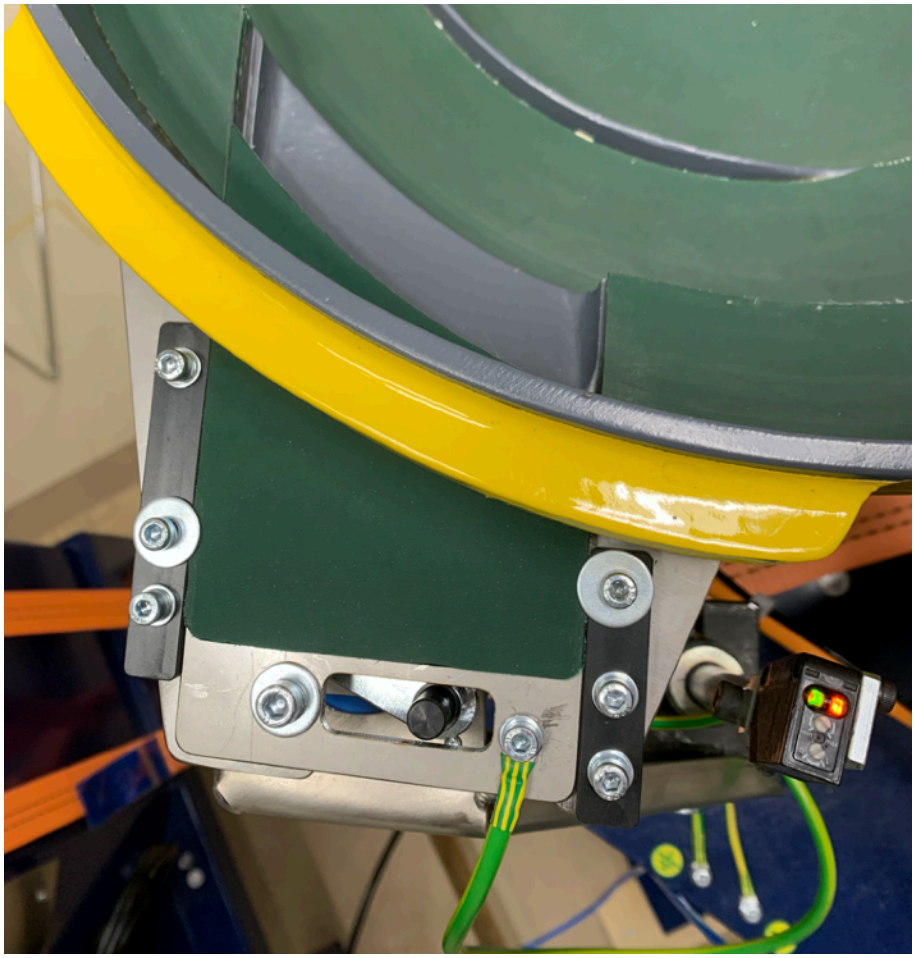


Figure 2.8: Horizontal scraper

2.2.2 Control system

The counting machine control system consists of the scrapers' control system, the vibrations control system and the belt conveyor control system. Every system is controlled separately and the efficiency of the counting machine depends on the settings of these three systems. The setting of each system must be done in consideration of the other two systems' settings and diameters of sorted LEGO elements. Every type of element has different settings in these three systems and the settings can change over time due to the wear of the green layer on the counting machine's paths.

3 OPC UA

Open Platform Communications Unified Architecture, abbreviated OPC UA, is the next generation of OPC technology developed by the OPC Foundation. The first version was released in 2006. OPC UA is a platform and manufacturer-independent standard that enables secure data exchange between machines, devices, and other industrial systems. These features make OPC UA one of the most important communication standards for Industry 4.0.

OPC UA supports Client/Server architecture. Data provided by Servers are transported over network to Clients using web services such as HTTPS, SOAP, or TCP and the transmission security is ensured by data encryption at the data source.

Security, transport, communication model and other specifications are designed to be scalable. This feature ensures that OPC UA is capable of supporting devices with different resource constraints.

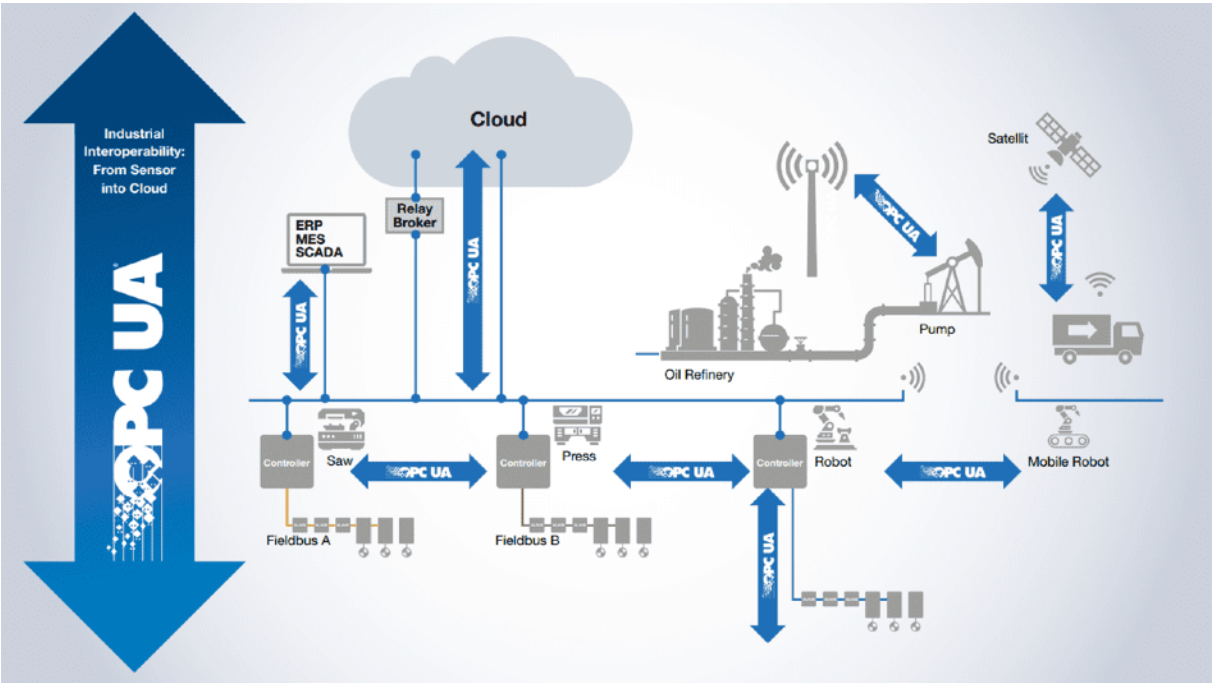


Figure 3.1: OPC UA data exchange structure [10]

A basic unit of data that allows OPC UA Servers to represent objects to OPC UA Clients is called a Node. A Node is described by attributes and relationships with other Nodes. Nodes that share the same attributes with other nodes are in the same Node Class, some of the core Node Classes are objects, methods and variables. [4] [5] [6] [7] [8]

3.1 UA vs. Classic

OPC Classic, previously known as OPC (Open Platform Communications), is the predecessor of OPC UA, released in 1996. Even though it enables data exchange between applications and automation devices, it has its limitations that do not meet the demanding requirements of today's industrial systems, such as support for complex data models or built-in security functions. The main disadvantage is that OPC Classic is not platform independent, instead it depends on the Microsoft Windows platform using COM and DCOM technologies for data exchange. Additionally, COM and DCOM technologies also cause OPC Classic vulnerability to attacks from viruses and malware. [4] [5] [6] [7]

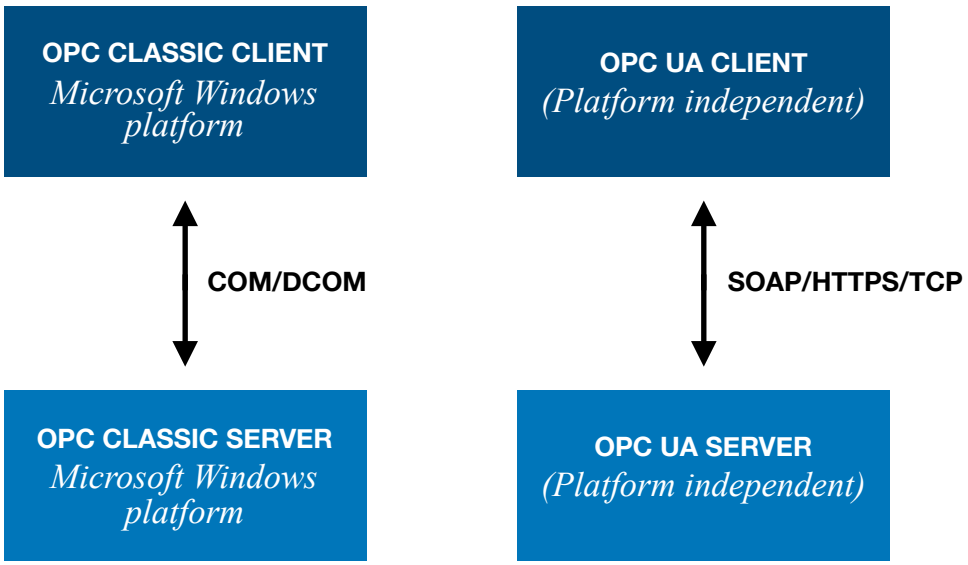


Figure 3.2: OPC UA vs. OPC Classic data exchange

3.2 Client

An OPC UA Client is a device that initiates a communication session by connecting to one or more end devices and controls them. The client sends message requests to the Servers and receives responses and notification requests from them. In the OPC UA Server/Client relationship, there is no restriction on how many Servers can be connected to one Client or how many Clients can talk to a single Server. Some Servers can even have an embedded Client, therefore they can function either as a Client, a Server or both.

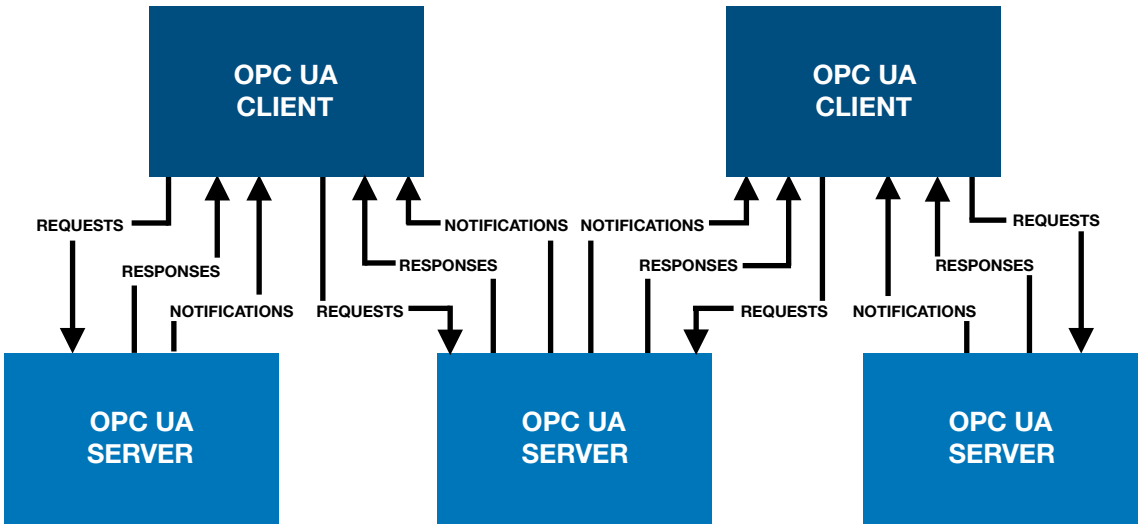


Figure 3.3: Client/Server relationship [4]

OPC UA Client contains a built-in Discovery process that is user-configurable. This process allows a Client to find and connect to Server devices without user intervention. The information about a Server can be obtained directly from the Server or by accessing a Discovery Server that catalogs available OPC UA Servers and provides Servers' information to the Client. [4] [7]

3.3 Server

Like other industrial Servers, OPC UA Server is an endpoint device that provides a physical interface to the outside world and operates under the direction of its counterpart – the Client. OPC UA Server can be a small sensor or a large technological device and there is no limitation to how many data points a Server can host.

OPC UA Server specific capabilities and characteristics are described by a Server Profile. A Profile provides information to other devices about which specific services are supported and what type of security is used. A Nano Server Profile is a least functional Profile containing a minimum set of functionalities. [4] [7]

Figure 3.4: OPC UA Server overview [4]

3.4 Subscription service

OPC UA provides a subscription service by which a Client can subscribe to a selection of Nodes. Unlike polling (reading information permanently), a subscription service notifies the Client is notified by a Server only when the subscribed Nodes change. A Client can subscribe to different types of information. This service reduces the amount of transferred data, thus accelerates the communication between Servers and Clients. [4] [7] [9]

4 Stepper motor replacement

A stepper motor's main disadvantage in this specific application is its lack of position feedback. By using a servomotor instead of a stepper motor, feedback information about the position can be provided and the whole process of adjusting the scrapers is simpler.

4.1 Motivation

Both stepper motors are each connected to one scraper and their rotational movement is converted to scraper's linear movement using a crank. Input for scrapers is a value in millimeters. For a vertical scraper, this value describes the space between the spiral track and the bottom of the metal part of the scraper (figure 4.1). For a horizontal scraper, the value describes the width of a part of the spiral path (figure 4.2). To set correct scrapers' positions in millimeters using the motors, the calibration needs to be done first. By the calibration, the value of the motors' angular positions is obtained. This value describes when the scrapers are in their minimal position, which is 0 millimeters. Getting this value using a stepper motor is complicated since the stepper motor does not have any feedback information about its current angular position. The calibration needs to be done semi-manually using multiple calibers which is time-consuming and cannot be done by the system itself but requires human interaction. By using servomotors, the need of any kind of manual work is eliminated and the calibration can be done quickly just by the system because of servomotors' current angular position feedback information. Another advantage of the servomotor is that positioning can be done to the specified position instead of by how much the motor should turn. Positioning to a specified position ensures that inaccuracies caused by the motor's inexact positioning will not increase over time.

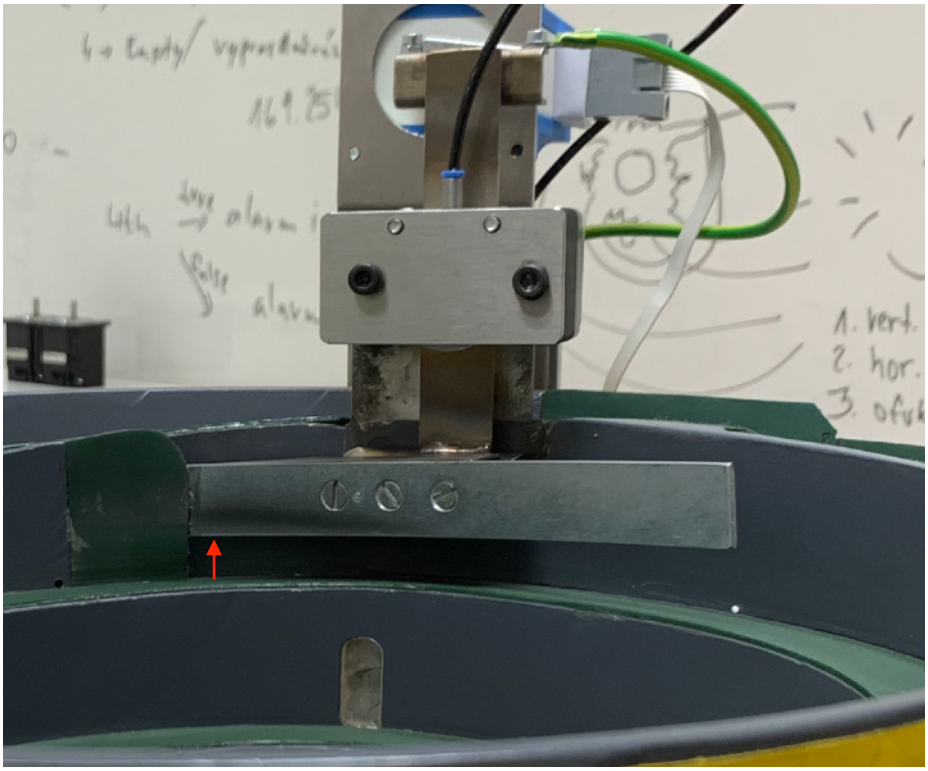


Figure 4.1: Vertical scraper setting value

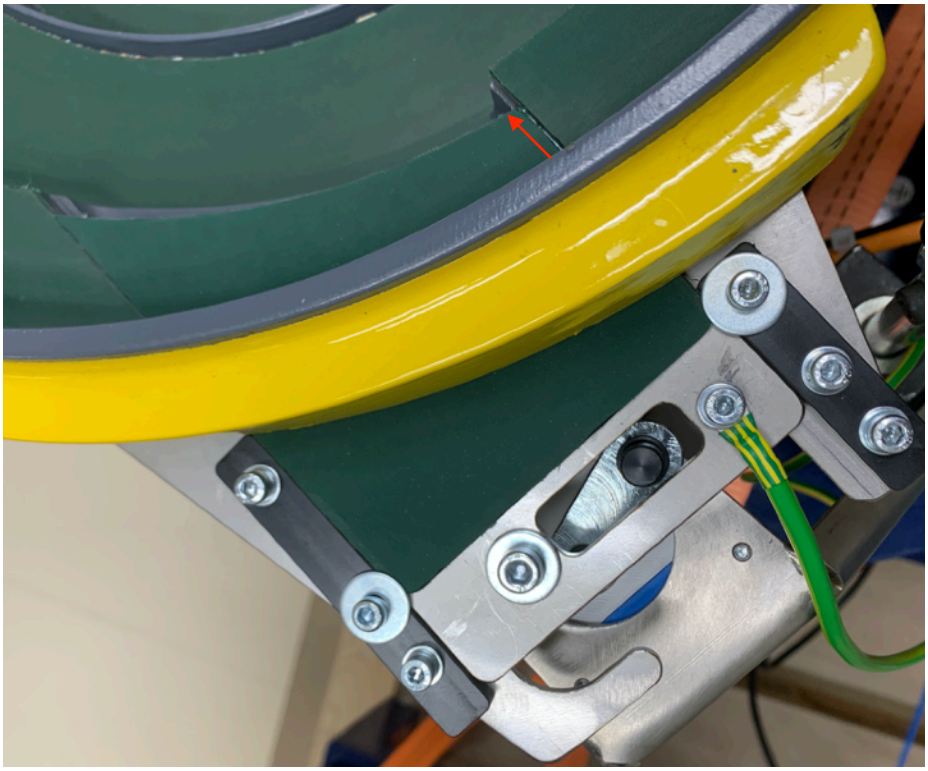


Figure 4.2: Horizontal scraper setting value

4.1.1 Stepper motor

A Stepper motor is a brushless DC motor that works in steps. That means a full rotation is divided into several equal sections that can be defined as individual steps. Stepper motor consists of a stator and a rotor. The stator is the stationary part, while the rotor is a shaft with a gear-like part performing a rotary motion. If there is a voltage input, stator creates a magnetic field due to the electromagnets that are distributed along the inner perimeter. Electromagnets are energized one by one, and the shaft performs rotational movement as the metal teeth are attracted by the magnetic force. Stepper motors often operate in open-loop systems along with a controller. The controller converts a series of input pulses into an angular displacement signal by which the motor is controlled. Stepper motors are precise in positioning and speed control, but the lack of a position sensor is causing no feedback information. [11] [12] [13]

4.1.2 Servomotor

A servomotor is a closed loop servomechanism that allows for precise control of either angular or linear position and speed. Servomotors can be rotary or linear, they are not a specific class of motors. A motor itself is a combination of two specific parts: a simple electric motor (AC or DC) and a sensor for position feedback. The output shaft of a servomotor can be rotated by a defined angle – an ability that regular motors do not have. Servomechanism uses negative feedback to control the correct position and speed. The sensor providing position and speed feedback is a type of encoder.
[14] [15] [16] [17]

4.2 LEGO Mindstorms

Lego Mindstorms is a programmable robot structure set developed by LEGO. Each set contains sensors, motors, a computer LEGO brick and LEGO elements. Since its launch in 1998, there were several series of LEGO Mindstorms sets.

LEGO motors will be used as a substitute for stepper motors. For controlling LEGO motors, a LEGO Hub is needed. A LEGO Hub and LEGO motors are taken from LEGO Mindstorms Robot Inventor 51515 kit. [18]

4.2.1 LEGO Hub

The LEGO Hub is a programmable control unit with a battery located at the back of the hub. Due to its power supply, it can work autonomously or in Streaming Mode with LEGO Mindstorms application. Using a USB Micro-B connector that is on the top side of the hub, you can recharge the battery or connect to other devices such as computers, phones or tablets. A connection with devices can also be established by Bluetooth, which can be turned on and off by a dedicated button on the front side of the hub. A 5x5 LED Matrix display and three buttons, which together form a light, and button control interface are also located on the front side. The hub contains 3 input/ output ports on both left and right side for motors and sensors to be attached to. [19]

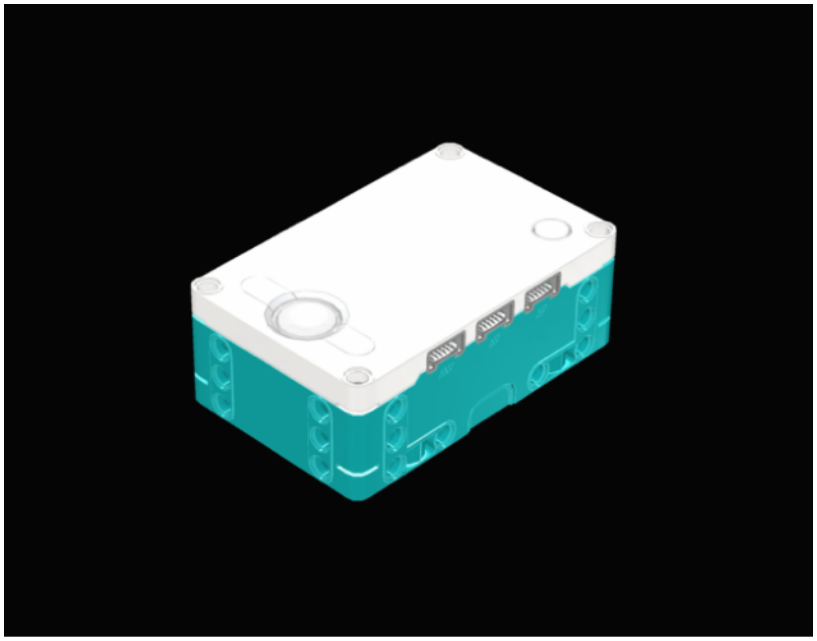


Figure 4.3: LEGO Hub [19]

LEGO Mindstorms application offers an option to create programs using word blocks or python programming language. Created programs can be uploaded and stored in the hub. In the autonomous hub mode, those programs can be run using three button interface – two arrow buttons to toggle between stored programs and the center button to run or stop selected program. While operating the buttons, the LED Matrix displays the program number that is currently selected and a *play* symbol indicates whether the program is running. Six-axis Internal Measurement Unit sensor allows the hub to measure the angle, orientation, gestures, acceleration, and regular velocity of the hub. This feature allows the hub itself to be a part of the created programs. [19]

Hub Features

- Front side: the display matrix, three buttons, Bluetooth button
- Back side: removable battery
- Left side: 3 input/output ports (A, C, E)
- Right side: 3 input/output ports (B, D, F)
- Bottom side: speaker
- Top side: USB Micro-B connector

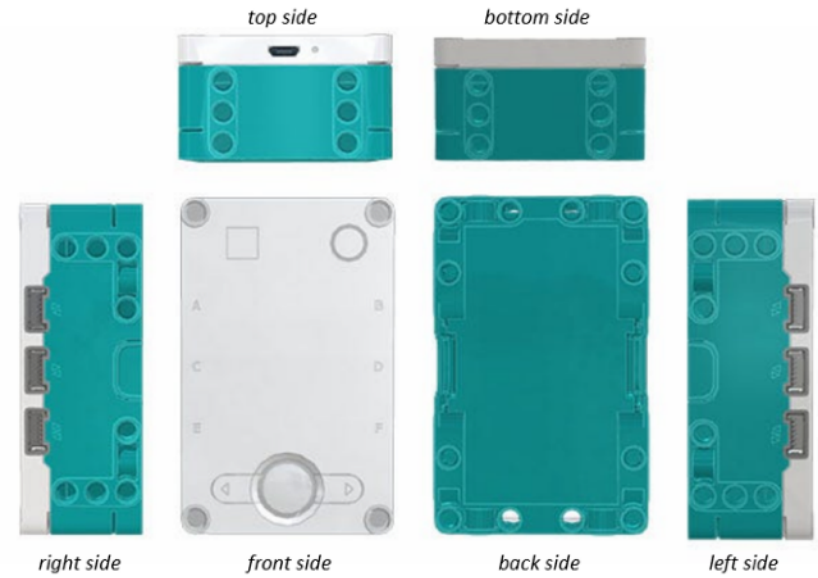


Figure 4.4: LEGO Hub overview [19]

4.2.3 LEGO motor

The LEGO motor is equipped with an integrated advanced rotation sensor that can report both speed and position values, so it can be considered a servomotor. It is a DC electric motor that contains gears for higher torques, therefore this relatively small and light motor has a lot of power for its size. The main target group for LEGO sets is children, so the motor is resistant to rotations by hand and can sense and measure this direct input. The motor can also be stalled without taking any damage and has a built-in function to detect it. The values in which the motor operates are integers, that means a position can be set and read only in full degrees. [19]

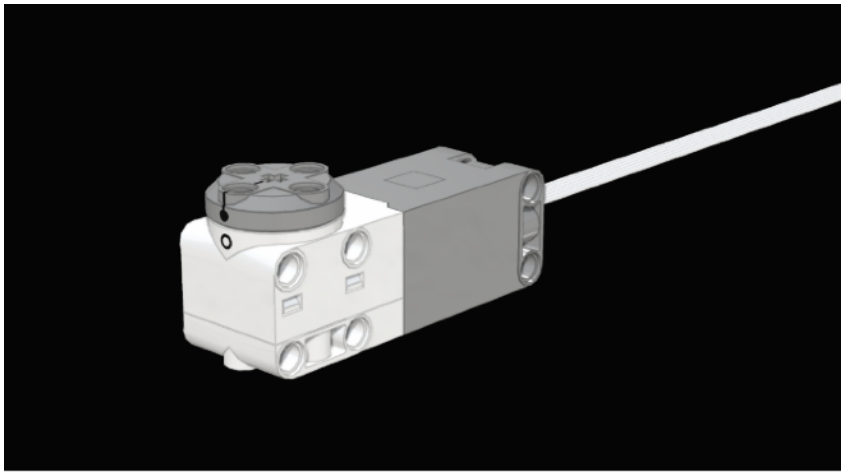


Figure 4.5: LEGO motor [19]

The motor recognizes two positions in degrees: absolute and relative. Absolute position is in the range of 0-359 degrees. Relative position can be any, but both must meet the condition of full degrees. Position reading can be done only in absolute degrees but setting motor position can be done in various ways. Using python library *mindstorms*, some basic functions for motor actions are:

run_to_position(): Runs the motor to an absolute position
run_for_degrees(): Runs the motor for a specified number of degrees
run_for_rotations(): Runs the motor for a specified number of rotations
run_for_seconds(): Runs the motor for a specified number of seconds
start(): Starts running the motor until it gets another command or a program ends

In each function above, the speed of the motor rotation can be set. The direction of rotation is given by the sign of the speed in all but the first example - *run_to_position()*. In this function, the direction is given separately by the keyword *'clockwise', 'counterclockwise',* or *'shortest path'*. It is also the only mentioned function that operates in absolute degrees. To make it easier for a user to estimate the absolute motor position just by their eyes, there is a zero-point marking on both the main body and the rotation part of the motor (figure 4.7). When both markings are aligned, the absolute position of the motor is zero, and the value increases in a clockwise direction from 0 to 359 degrees. [19]

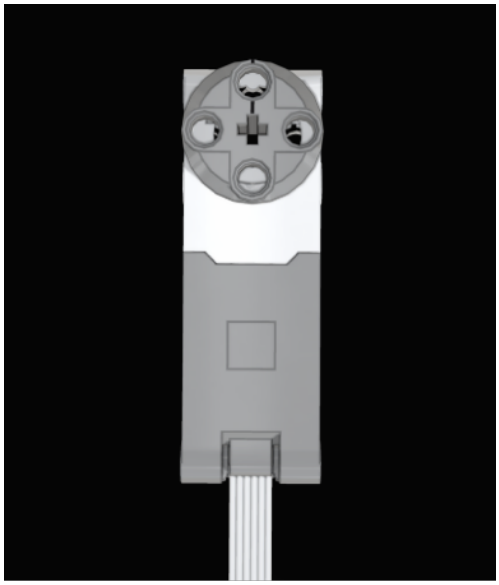


Figure 4.6: LEGO motor - front view [19]

Figure 4.7:Zero-point markings [19]

4.3 3D Printing

In the previous solution, both stepper motors were attached to the counting machine using a metal sheet and 4 screws (figure 4.10 and 4.11). Since the shape and dimensions of the stepper motor are different from those of the LEGO motor, LEGO motors can not be attached to the counting machine the same way as stepper motors.

For motor mounting, a 3D printed enclosure was used. This solution enables a design of the enclosure concerning the other dimensions and ensures that no other components have to be changed or customized. In addition to the enclosure, a shaft was also 3D printed because it is needed for connecting the motor with the crank and the LEGO motor does not have one.

As shown in figure 4.5, the LEGO motor does not have any kind of mounting holes except for the LEGO holes. LEGO element 2780 has a connecting function in LEGO sets and fits the LEGO holes.

Figure 4.8: LEGO 2780 [20]

By obtaining the dimensions of the LEGO holes (figure 4.9), holes can be included in the 3D printed models and used along with LEGO elements 2780 for attaching 3D printed objects to a LEGO motor. Models were made in Autodesk Inventor Software and 3D printed on Original Prusa i3 MK3S+.

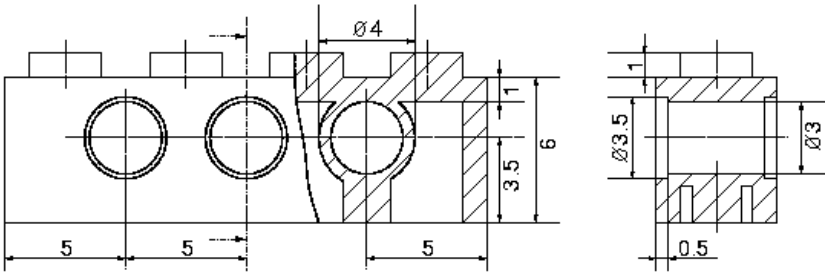


Figure 4.9: LEGO holes dimensions [21]

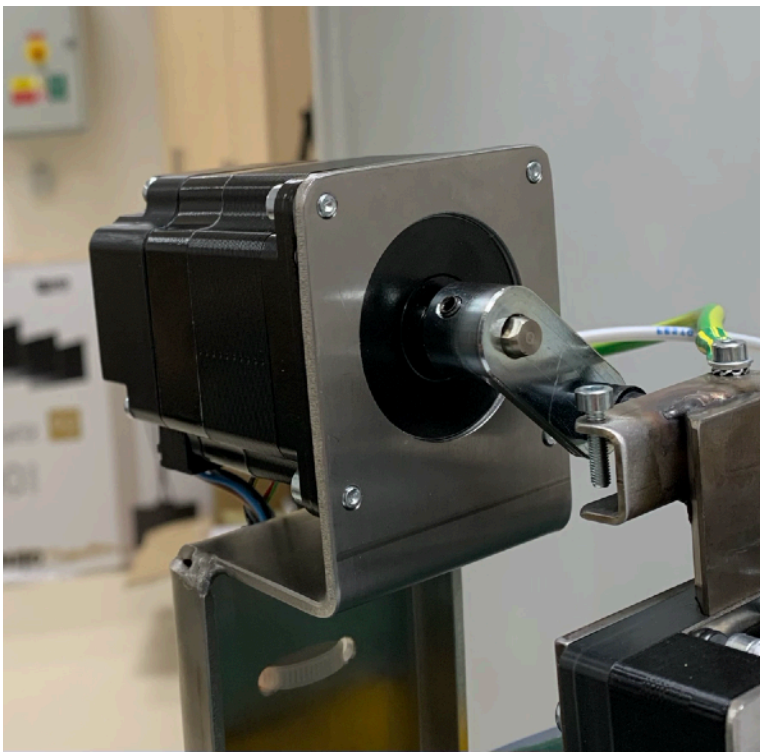


Figure 4.10: Stepper motor for vertical scraper

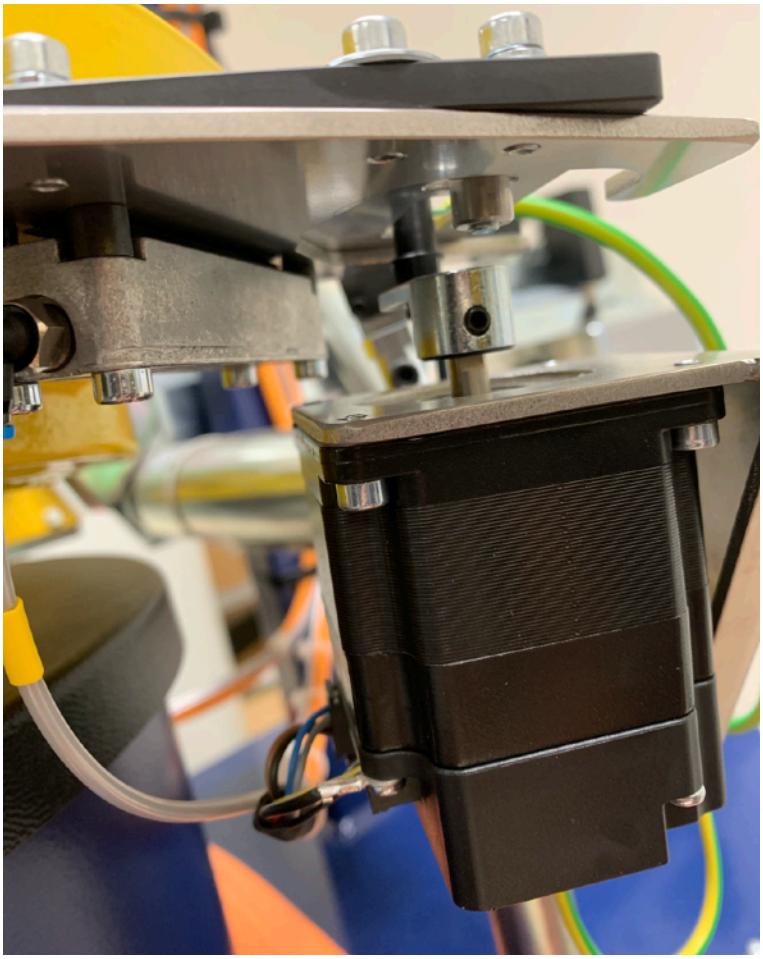


Figure 4.11: Stepper motor for horizontal scraper

4.3.1 Shaft for LEGO motor

The shaft (figures 4.12 A-D) is 3D printed with 4 LEGO holes for attaching it to the motor using 2780 LEGO elements. The end part of the shaft is flat, so the crank could be attached and does not twist. In figures 4.13 A and 4.13 B, a 3D printed shaft attached to the LEGO motor is shown.

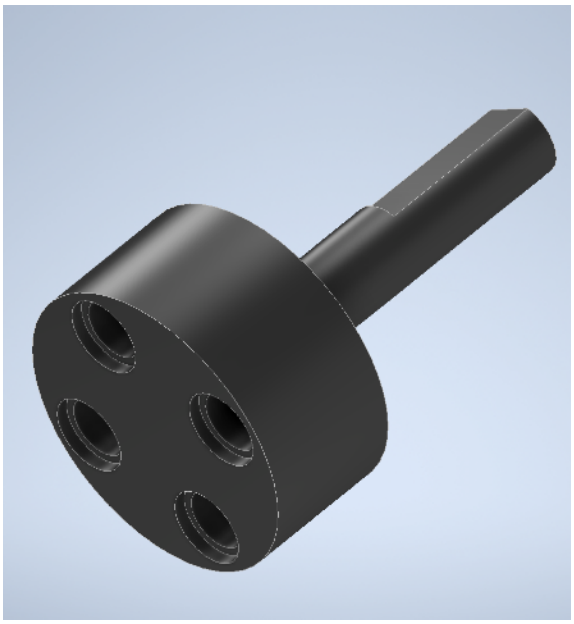


Figure 4.12 A: Shaft for LEGO motor

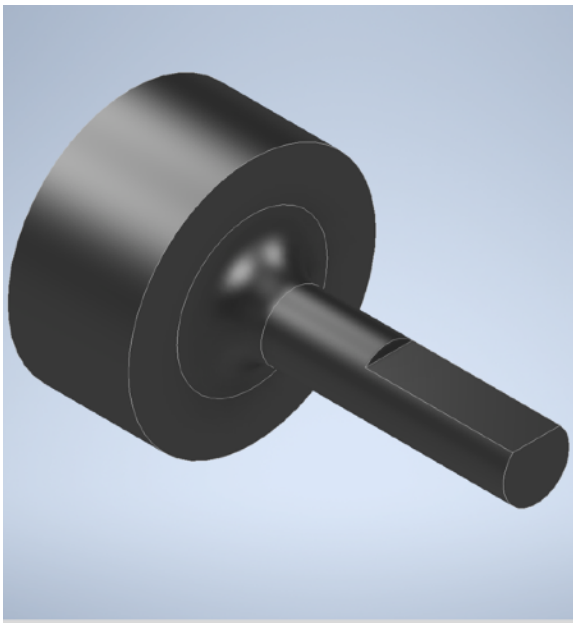


Figure 4.12 B: Shaft for LEGO motor

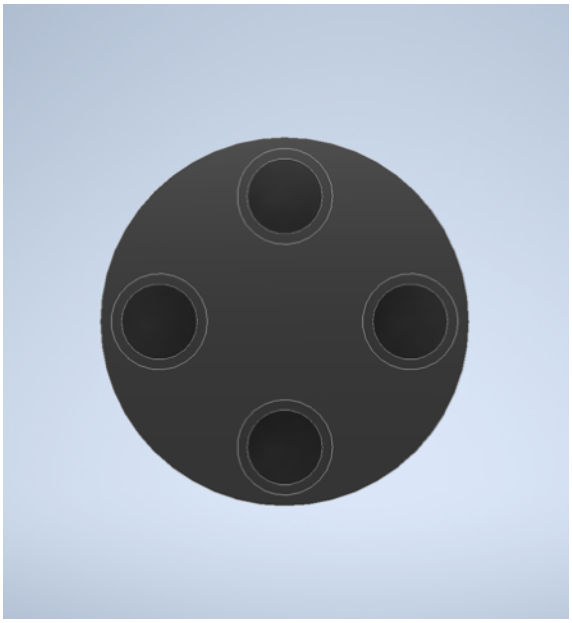


Figure 4.12 C: Shaft for LEGO motor

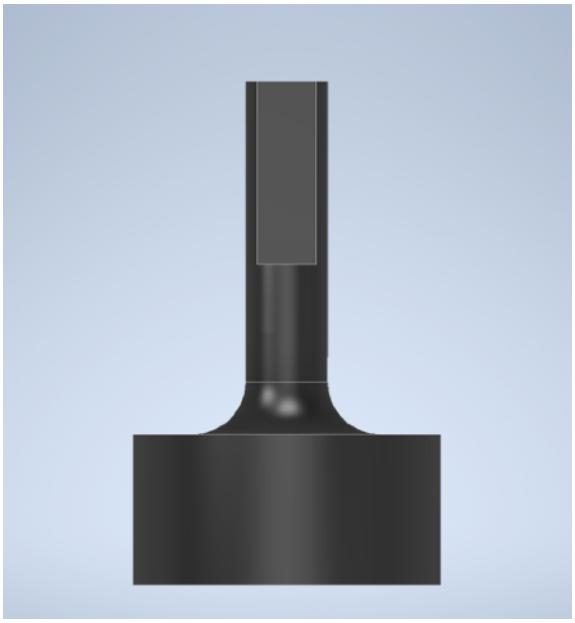


Figure 4.12 D: Shaft for LEGO motor

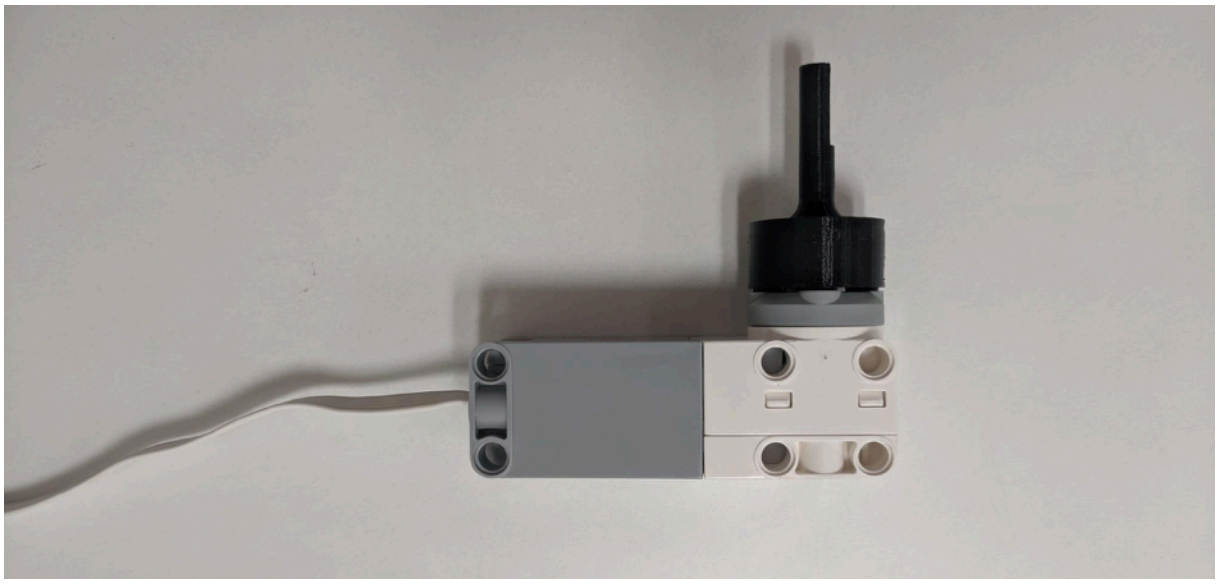


Figure 4.13 A: Shaft attached to LEGO motor

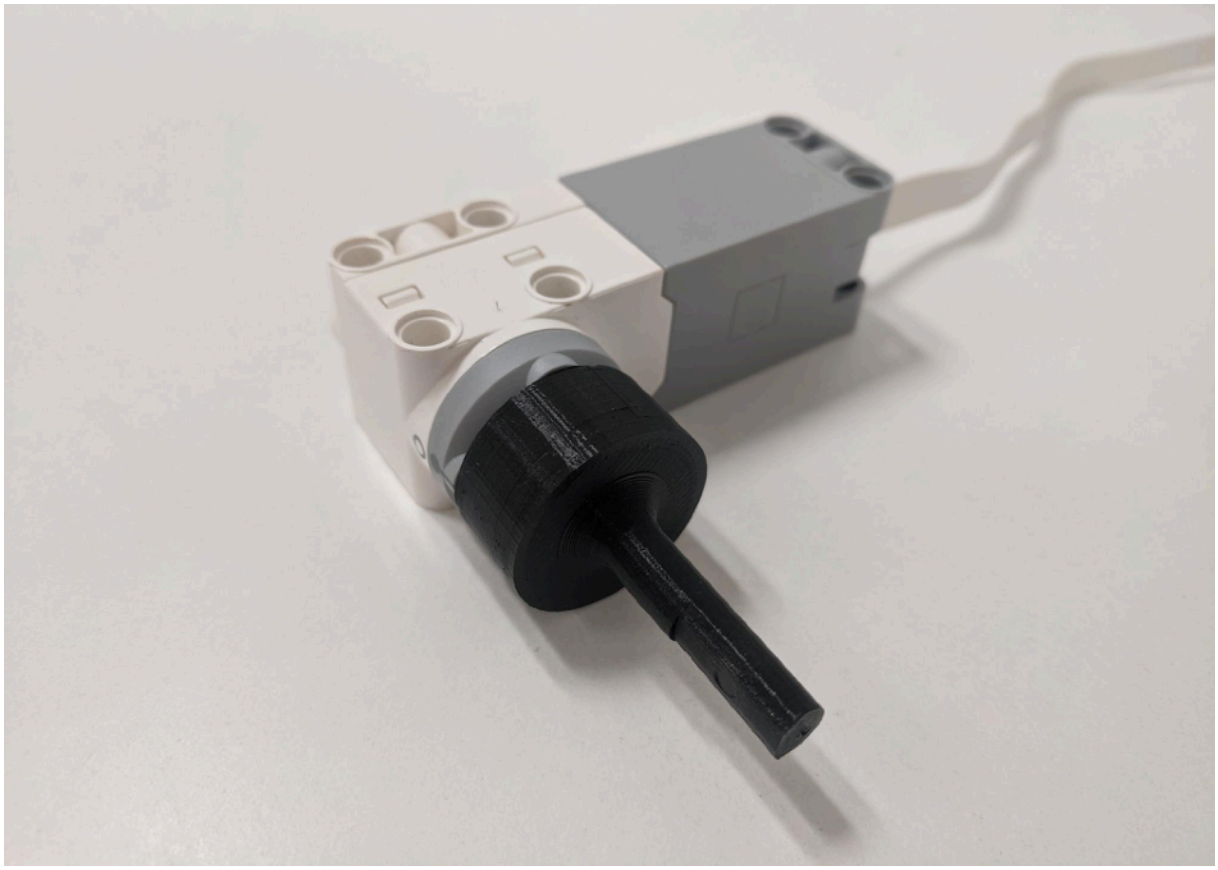


Figure 4.13 B: Shaft attached to LEGO motor

4.3.2 Enclosure for LEGO motor

The enclosure for the LEGO motor consists of two parts that fit together. The motor with the shaft is inserted into the first part (figures 4.14 A-D). The shaft is stretched through a hole into which a bearing is inserted.

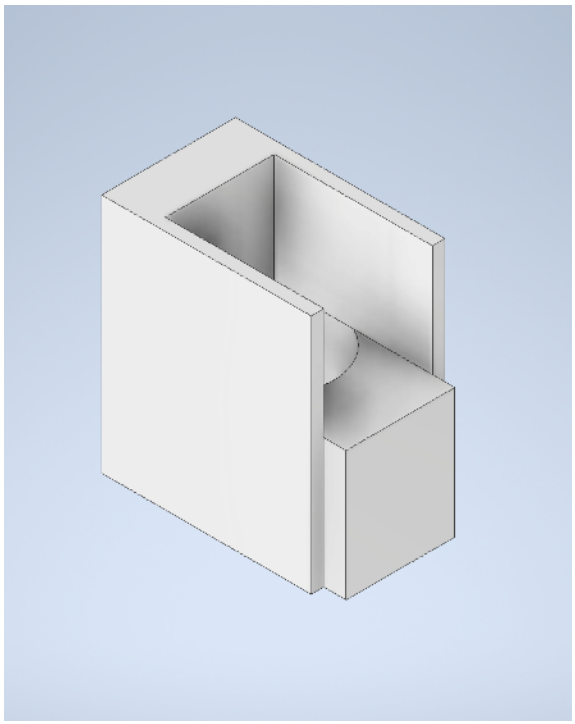


Figure 4.14 A: Enclosure for LEGO motor

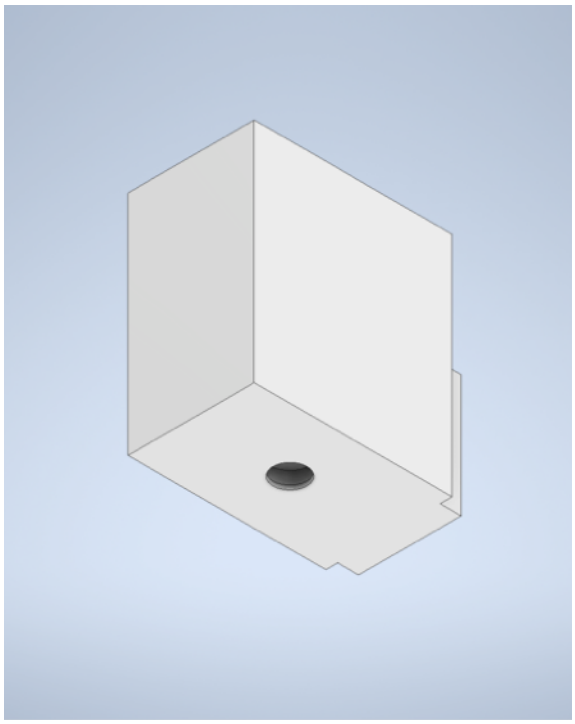


Figure 4.14 B: Enclosure for LEGO motor

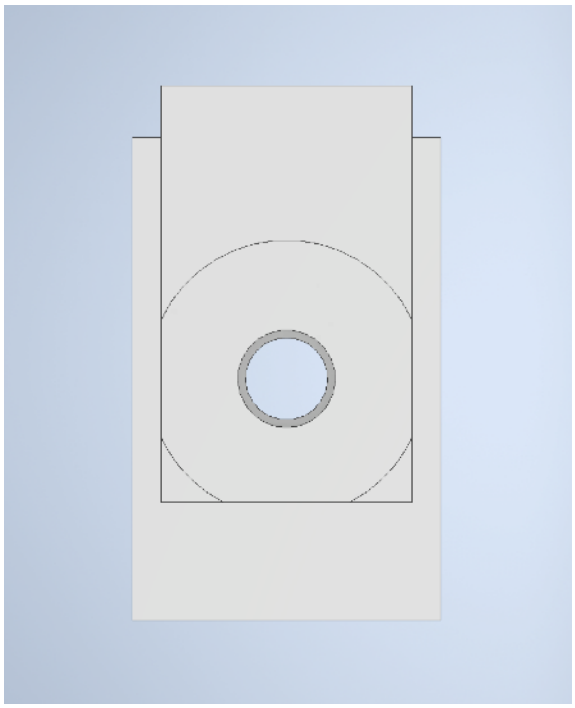


Figure 4.14 C: Enclosure for LEGO motor

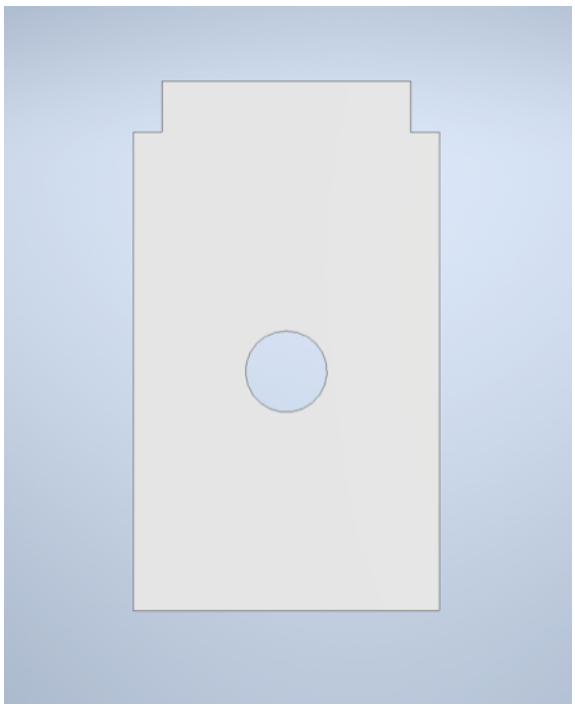


Figure 4.14 D: Enclosure for LEGO motor

The first part with the motor in it is inserted into the second part (figures 4.14 E-G). The second part has 3 LEGO holes for attaching the LEGO motor and 4 screw holes that are used to mount the enclosure with the motor to the counting machine. The whole enclosure with LEGO motor mounted to the counting machine is shown in figures 4.15 A-B for the vertical scraper and in 4.16 A-B for the horizontal scraper.

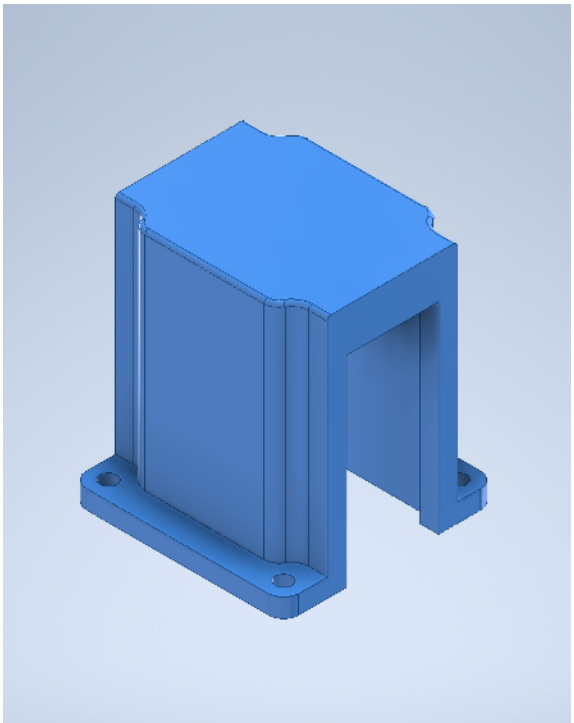


Figure 4.14 E: Enclosure for LEGO motor

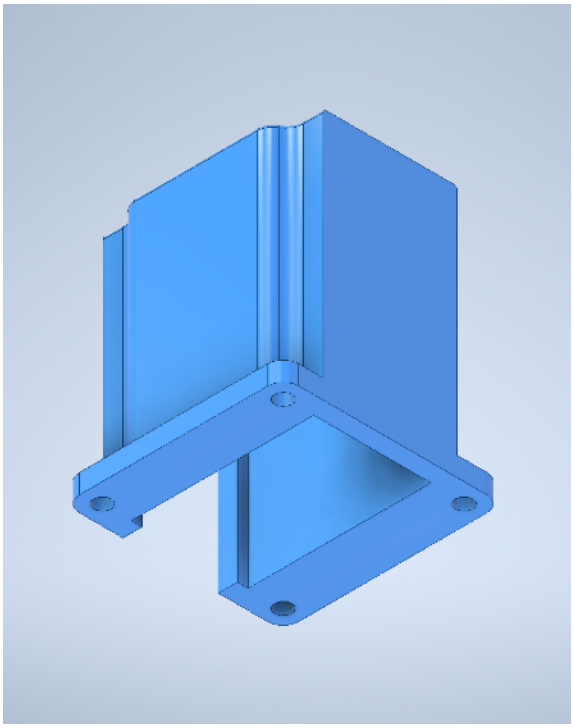


Figure 4.14 F: Enclosure for LEGO motor

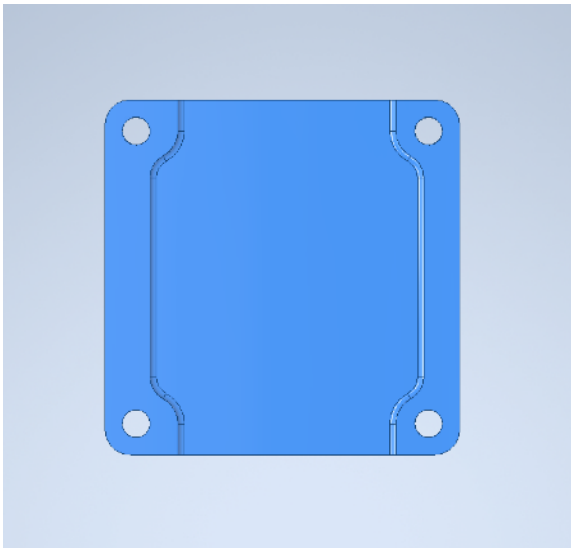


Figure 4.14 G: Enclosure for LEGO motor

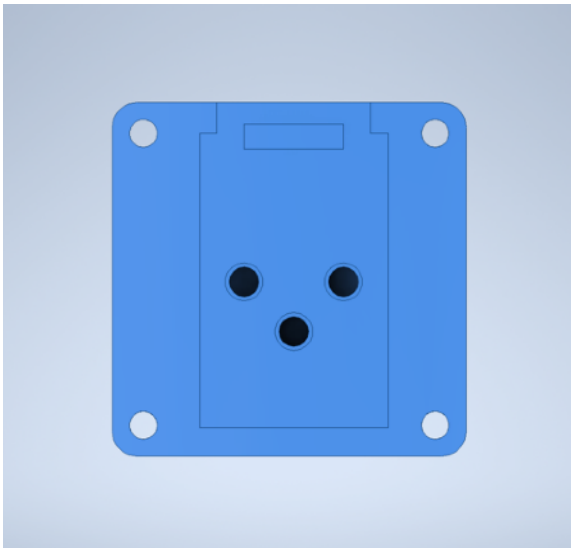


Figure 4.14 G: Enclosure for LEGO motor

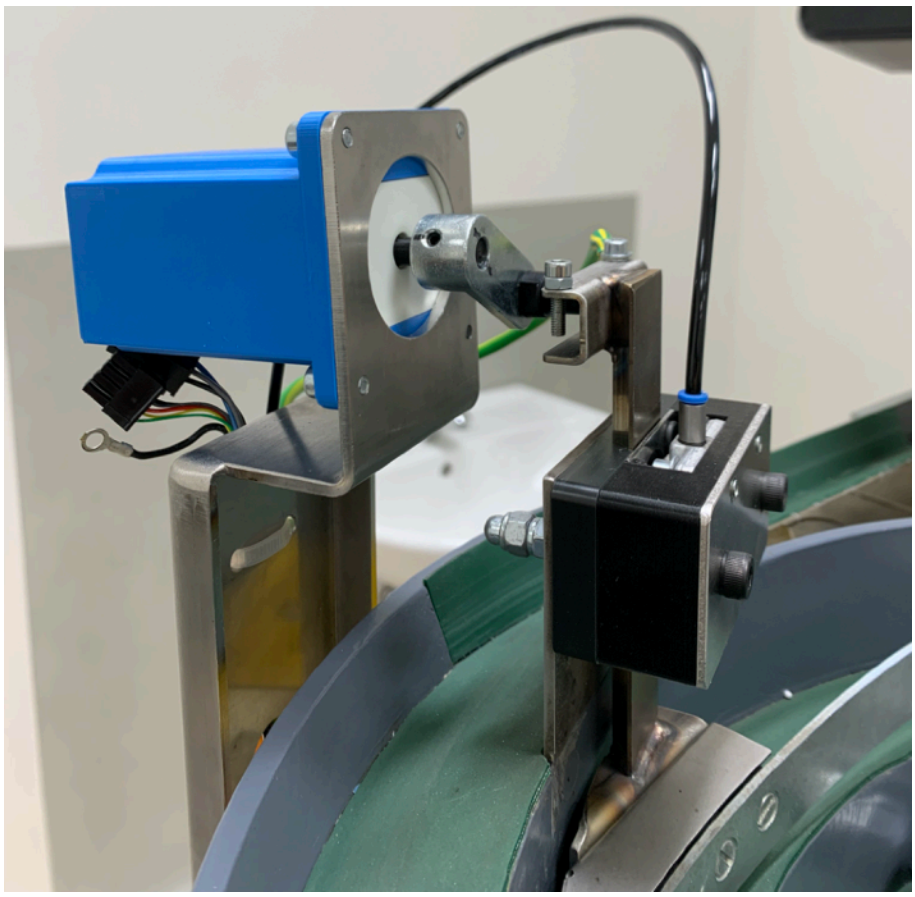


Figure 4.15 A: Enclosure for LEGO motor - vertical scraper

Figure 4.15 B: Enclosure for LEGO motor - vertical scraper

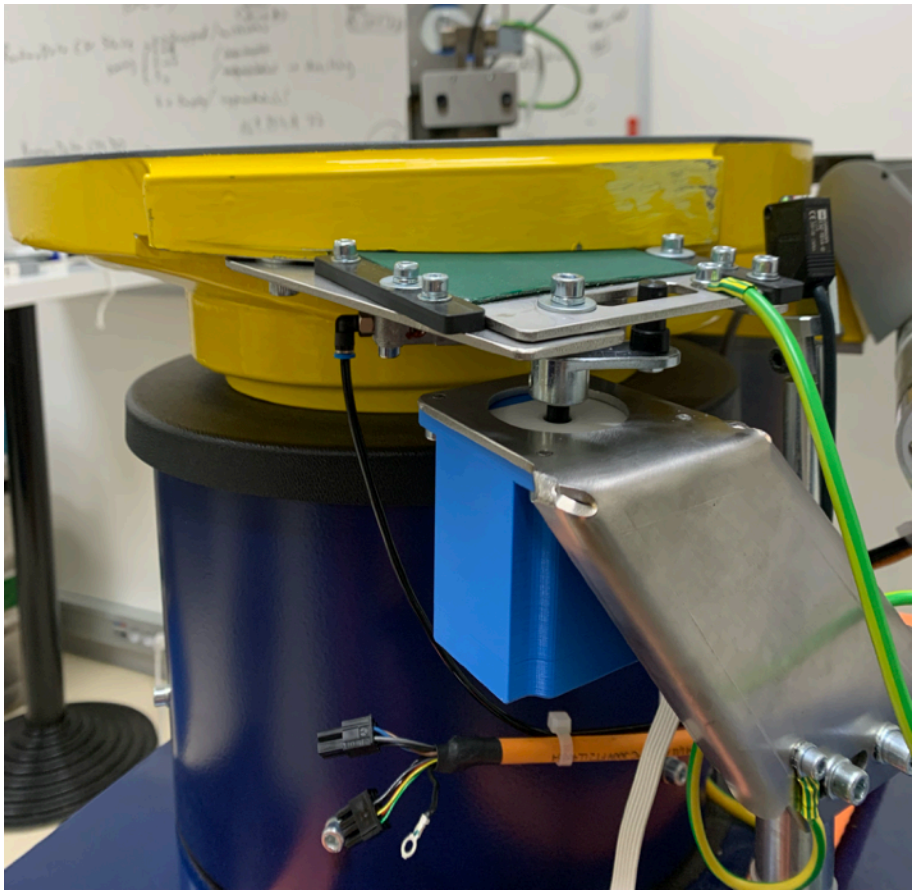


Figure 4.16 A: Enclosure for LEGO motor - horizontal

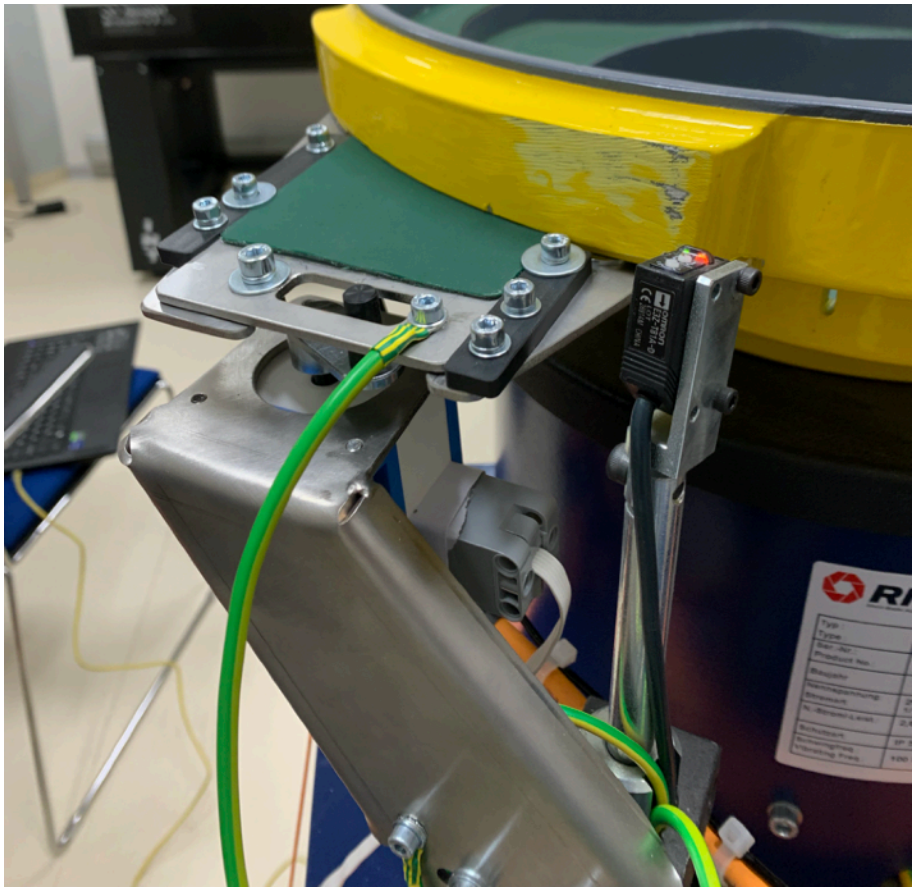


Figure 4.16 B: Enclosure for LEGO motor - horizontal

5 Scrapers' control system

The scraper control system can be divided into two main parts: the pneumatic system and the motors. The previous solution for scrapers' control was built on PLC. The PLC controlled both: the motors and the pneumatic system. The new solution is built on Raspberry Pi using Python programming language.

The pneumatic system is controlled by two pneumatic valves. Adjustment of the scrapers is secured by two LEGO motors. The Raspberry Pi can not provide direct control of either of those two systems. Therefore the LEGO motors are controlled by a a LEGO Hub and the pneumatic valves are controlled by an Automation Hat. Both LEGO Hub and Automation Hat are controlled by Raspberry Pi. The OPC UA Server that is created within the control script secures the communication with the existing OPC UA Client.

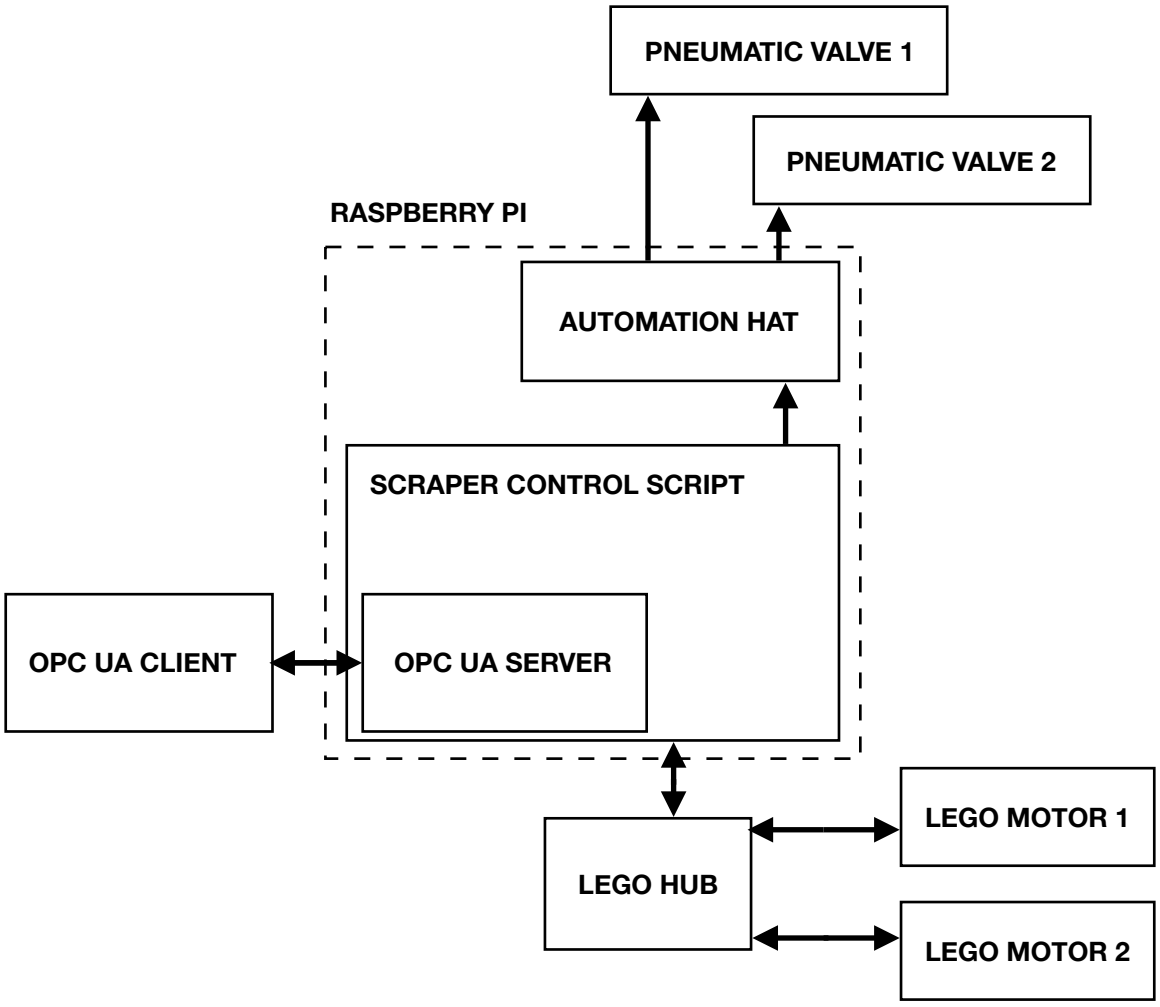


Figure 5.1.: Scrapers' control systems

5.1 Pneumatic system control

The scrapers' control system includes the pneumatic brakes and the air-flow system, all controlled by solenoid direction control valves. Solenoid valves are electro-mechanically operated valves and direction control means that the valves can be described by several ports and states determining in which directions and in how many ways the fluid can flow. Those valves' characteristics are mostly described by pneumatic symbols for valves.

While the horizontal scraper only has a pneumatic brake, the vertical scraper has both a pneumatic brake and an air-flow system. Both pneumatic brakes are controlled by solenoid valve - Festo VUVG-L10-T32U-AT-M5-1P3 (figure 5.2) and valve - Festo VUVG-LK10-M52-AT-M5-1H2L-S (figure 5.3) controls the air-flow system. Both valves are direct current powered and their operating voltage is 24V. [22] [23] [24] [25]

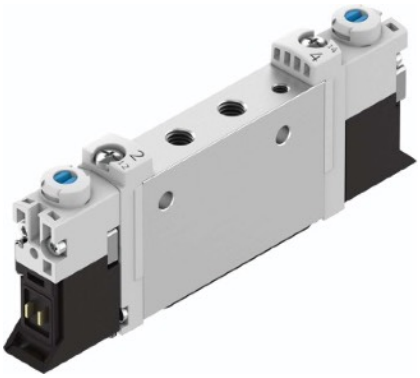


Figure 5.2: Pneumatic brakes valve [24]

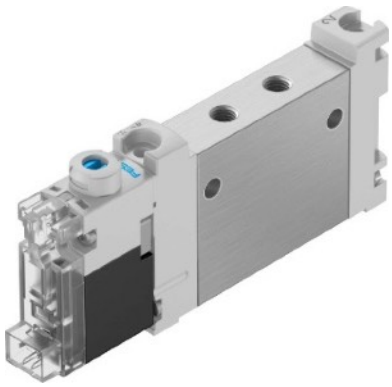


Figure 5.3: Air-flow system valve [25]

5.1.1 Pneumatic brakes and air-flow system

The valve for pneumatic brakes has 5 ports and 4 states as shown on the valve symbol (figure 5.4). The valve is operated by two power supply cables – each power supply cable operates one block and can switch it between two states. Air pressure enters port 1 and continues through ports 2 and 4, both blocks are in an un-actuated state so when there is no power input, both brakes are engaged. Initializing the power supply through both cables causes switching to actuated states. Port 1 is blinded in this state, meaning there is no air pressure in the system. Port 2 is now connected to port 3 and port 4 is connected to port 5. Ports 3 and 5 are exhausts, meaning trapped compressed air is vented out from the system through them and both brakes are released.

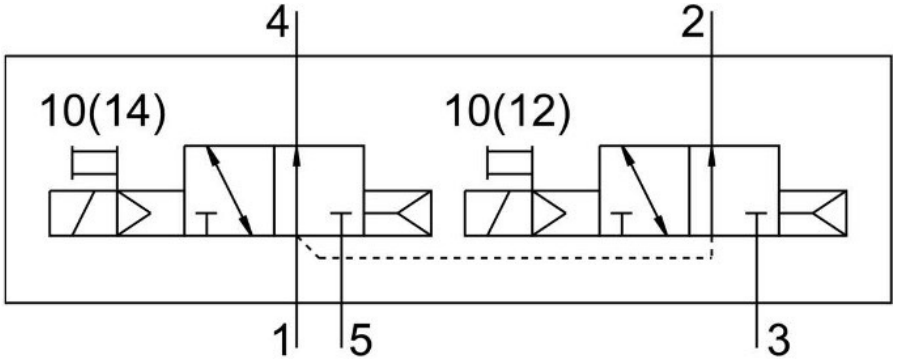


Figure 5.4: Pneumatic brakes valve symbol [24]

Air-flow system is connected to a valve with one power supply cable. This valve also has 5 ports but only 2 states, as shown in figure 5.5. Air pressure enters the valve through port 1 and continues to port 2 if the valve is in un-activated state. Port 2 is blinded, therefore without power input, there is no airflow. Switching to the second state by initializing the power supply, port 1 connects to port 4 and airflow is secured. Ports 5 and 3 are exhausts.

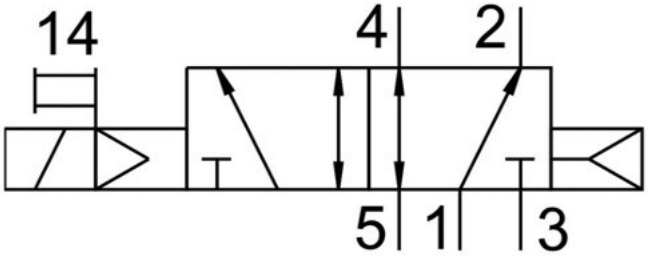


Figure 5.5: Air-flow system valve symbol [25]

5.1.2 Automation Hat

In the previous scrapers' control solution, the solenoid valves were controlled by PLC. In this solution, controlling them by PLC is no longer an option. The whole control script runs on Raspberry Pi, but the pins on Raspberry Pi do not support the required voltage and current on which solenoid valves operate. The solution to this problem is the Automation Hat by PIMORONI. This Automation Hat is compatible with 40-pin Raspberry Pi models and features relays, analog channels, powered outputs, and buffered inputs.

The automation HAT features three 24V relays that can each tolerate up to 2A. Three relays are just enough for three power supply cables that are needed to power up the valves. By using the PIMORONI Python library relay control is secured by one simple function: *relay.numberofrelay.on/off()*.

Vertical scraper's pneumatic brake is controlled by relay number one, horizontal scraper's pneumatic brake is controlled by relay number two, and relay number three controls the air-flow system. [26]

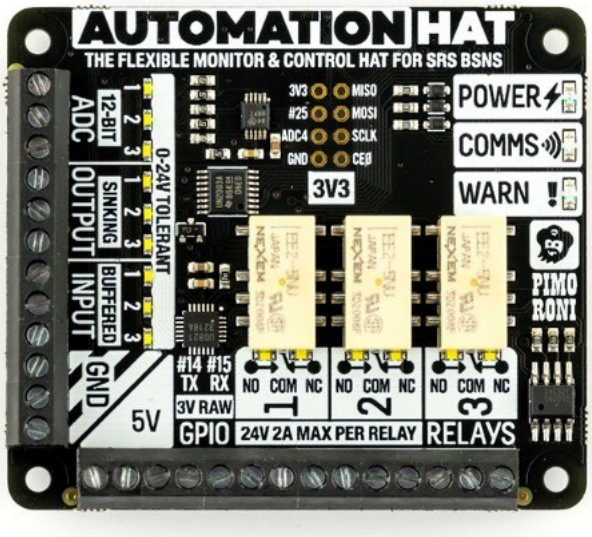


Figure 5.6: PIMORONI Automation Hat [26]

5.2 LEGO motor control

A LEGO Hub has a Python implementation called MicroPython. MicroPython that runs on a LEGO Hub is extended by a Python library called *mindstorms*. This Python library has been developed by LEGO and is specifically designed to provide access to motors, sensors, and the Hub. With the LEGO Mindstorms Python library, controlling motors is intuitive and simple since it is designed for young clientele.

A complication occurs when LEGO motor needs to be controlled from the Raspberry Pi control script. LEGO motor can not be connected to a Raspberry Pi, therefore connecting it directly is not an option. As described in 4.2.1, a LEGO Hub, which the LEGO motor can be connected to and operated by, has a USB Micro-B connector. This means, that a connection over the serial line can be established between a Raspberry Pi and a LEGO hub.

Controlling motor from Raspberry Pi through LEGO Hub using the serial line consists of sending string commands over the serial line that are executed in the LEGO Hub. By reading from serial line, the position feedback from the LEGO motor is secured. [27]

5.3 Scraper control

The main parts of the new solution for scraper control are the pneumatic valves controlled by the relays on the Automation Hat and the LEGO motors controlled by the LEGO Hub. The LEGO Hub is connected to a Raspberry Pi using the serial line. LEGO Hub controls motors and relays control valves so serial line and Automation Hat are used to connect both main parts to the control script that runs on Raspberry Pi since scraper control needs both to function well.

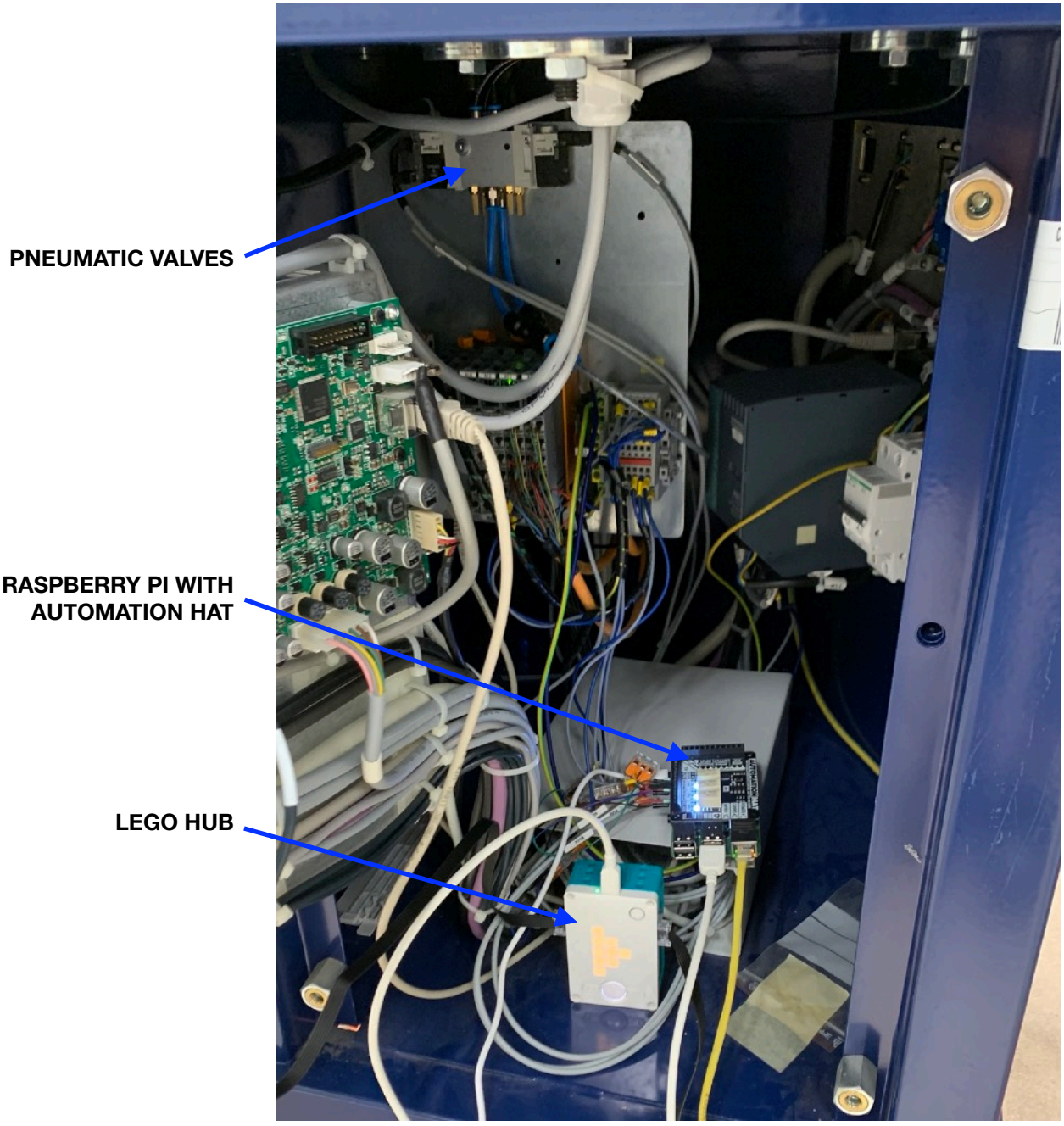


Figure 5.7: Scrapers' control system hardware

5.3.1 Recalculation

Scrapers' positions are set in millimeters and are defined and explained in 4.1. The LEGO motors are set by an absolute position in degrees. To set the correct scraper position in millimeters by LEGO motors, which take their absolute position in degrees as input, a recalculation must be made. Recalculations are also done from degrees to millimeters since there is feedback information from the motor about its position. Control script and recalculation work only when both motors are mounted correctly. The motor zero line in the figures describes a position where the motor's absolute position is zero degrees.

As shown in figure 5.8, the first part of the recalculation is from motor displacement in degrees to motor displacement in millimeters or vice versa. This recalculation uses the length of the crank, which is a constant, and motor displacement in degrees or motor displacement in millimeters depending on which of these two is a known value and which of these two is a required value. Since the result triangle is rectangular, a simple sine function is used for the first part of the recalculation.

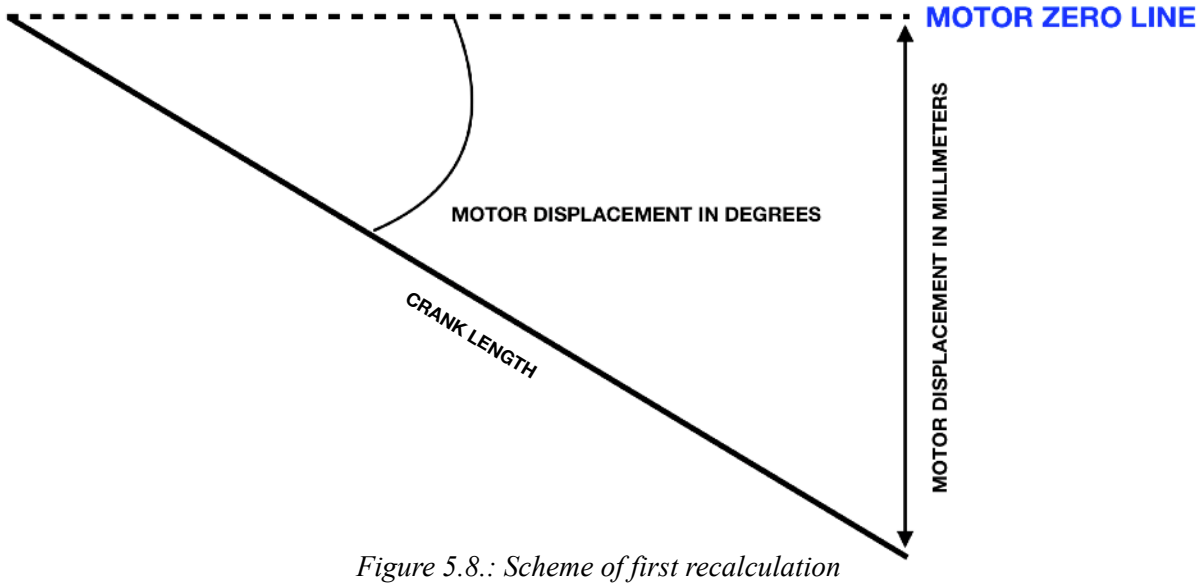


Figure 5.8.: Scheme of first recalculation

The second part of the recalculation is from motor displacement position in millimeters to scraper set position which is also in millimeters (figure 5.8). For this recalculation, the value from calibration is needed - this value describes the distance in millimeters between the motor zero line and the scraper zero line. When the calibration value is recalculated by the first part of the recalculation, the result value would be an angle by which a motor must rotate from its zero position to reach the minimal scraper position. The second recalculation is then done by simple subtraction of the known value from the calibration value (figure 5.9).

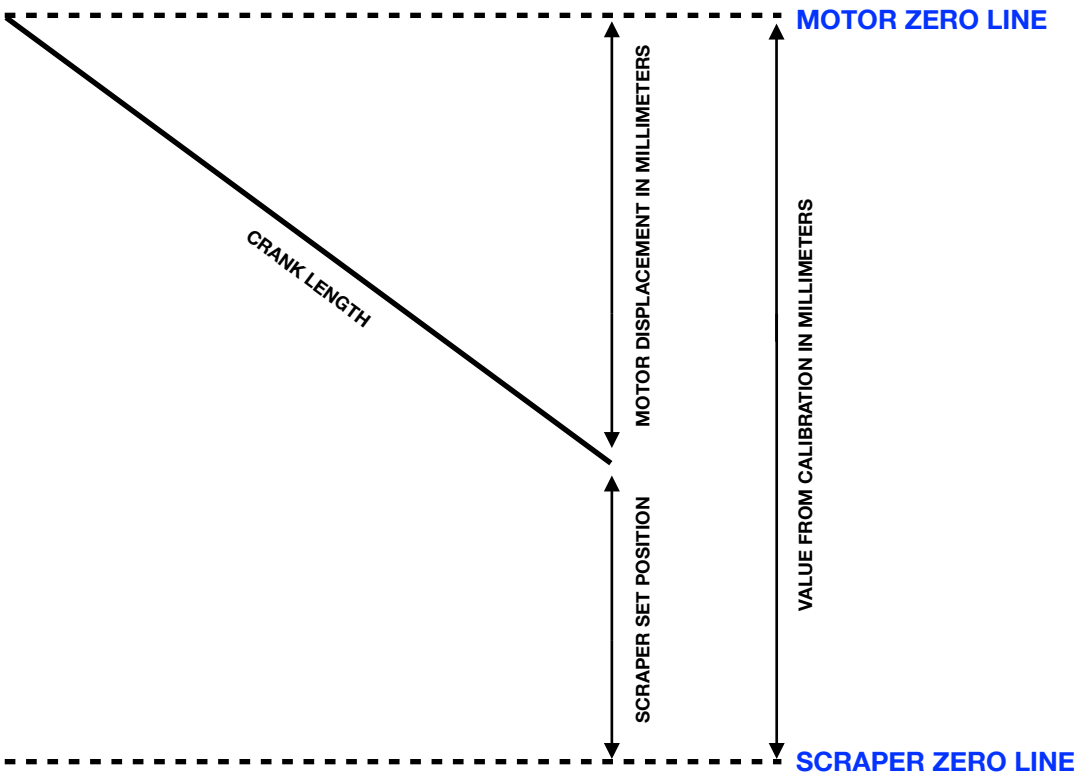


Figure 5.9.: Scheme of second recalculation

The third part of the recalculation introduces the *diff* constant. The *diff* constant is the result of the difference between the width of the slit and the diameter of the pin. This constant is used whenever the scraper is adjusted from the different side than the calibration was done.

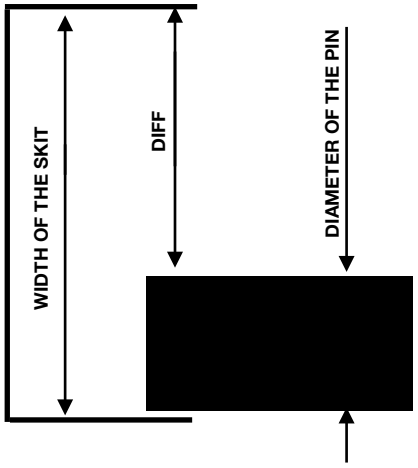


Figure 5.10.: Scheme of third recalculation

5.3.2 Calibration

Without calibration, scrapers can not be set to correct positions. In the scraper control script, there is an algorithm for calibration that returns a z_m variable. Variable z_m is recalculated to millimeters from variable z_d ,to which a value of how much the motor must turn to reach the scraper's zero position is assigned.

```
1  Function Calibration(p):
2      s ← "CLBRTN"+ p;
3      z_d ← 0;
4      if p equals p_h then
5          r ← relay number two;
6      else
7          r ← relay number one;
8      end
9      turn r on;
10     wait for 0.5 seconds;
11     write command on serial line to start motor on port p;
12     wait for 5 seconds;
13     write command on serial line to print s + position of motor on port p;
14     turn r off;
15     wait for 0.5 seconds;
16     while z_d = 0 do
17         if there is something on serial line then
18             l ← value on line;
19             if first 7 symbols of l = s then
20                 z_d ← value after seventh symbol of l;
21             end
22         end
23         wait for 0.1 seconds;
24     end
25     z_m ← recalculated z_d;
26     return z_m
```

Figure 5.11.: Pseudo-code for calibration

The pseudo-code for Calibration is shown in Figure 5.11 and takes *p* as the input. *p* in the control script expresses a port label to which a motor, that adjusts the scraper that is required for calibration, is connected. In line 2 a variable *s* is initialized and represents a string consisting of letters CLBRTN and a port label. If the condition in line 4 is met, relay two that controls the pneumatic brake of the horizontal scraper is assigned to variable r. Otherwise relay one that controls pneumatic brake of the vertical scraper is assigned to the *r* variable. The assigned relay is turned on and it releases the break in line 9. Wait time in line 10 is applied because the pneumatic system has a little delay. In line 11, there is a command to start running the motor that is connected to port *p* over the serial line. Wait time in line 12 ensures that the motor reaches the minimal scraper position before the command that reads the current motor position is sent over the serial line in line 13. The assigned relay is turned off and engages the break in line 14. The while loop in line 16 reads from the serial line until the if condition in line 19 is met and variable z_d reassignes its value from zero to the motor position in absolute degrees. In line 25, the value z_d is recalculated to millimeters.

5.3.3 Vertical scraper

The vertical scraper control must always be adjusted from the bottom up. The reason for that is that when the brake is released, the vertical scraper almost always falls a little on the pin of the crank. Therefore, the motor adjusts the scraper by the upper part of the slit. However, on rare occasions, the scraper gets stuck and when the brake is released, the scraper does not fall on the pin but stays in the same position. If the new set position of the scraper is lower than the current one, the outcome position of the scraper would be different when it is pulled by the lower part of the slit than by the upper part of the slit. The reason for that is, that the script controls a motor position and can not define by which part of the slit the scraper is being pulled.

As already mentioned, the solution is to always adjust the scraper from the bottom up. The control of the vertical scraper can be divided into three cases according to the relationship between the new and the current scraper position. All three cases start by releasing the brake and end by engaging the brake, reading the motor position, setting the new position as a current one and setting the motor to its neutral position.

The first case is when the new position is smaller than the current position and slit's width. In this case, the vertical scraper adjustment is divided into two parts. In part one, the scraper runs to its minimal position - 0, and in part two, the scraper is adjusted to the new required position from the bottom up. The second case is when the new position is higher than the current one. In this case, the vertical scraper is always adjusted from the bottom up. The third case is when the new position is smaller than the current position but higher than slit's width. In this case, the vertical scraper adjustment is also divided into two parts, as in the first case. In part one, the scraper is adjusted to the new position minus the slit's width. In part two, the scraper is adjusted to the new position from the bottom up.

Since the calibration is made from a direction different from the vertical scraper's adjustment direction, when setting the motor's position, the *diff* constant must be added to the new position of the scraper. The *diff* constant is explained better in 5.3.1.

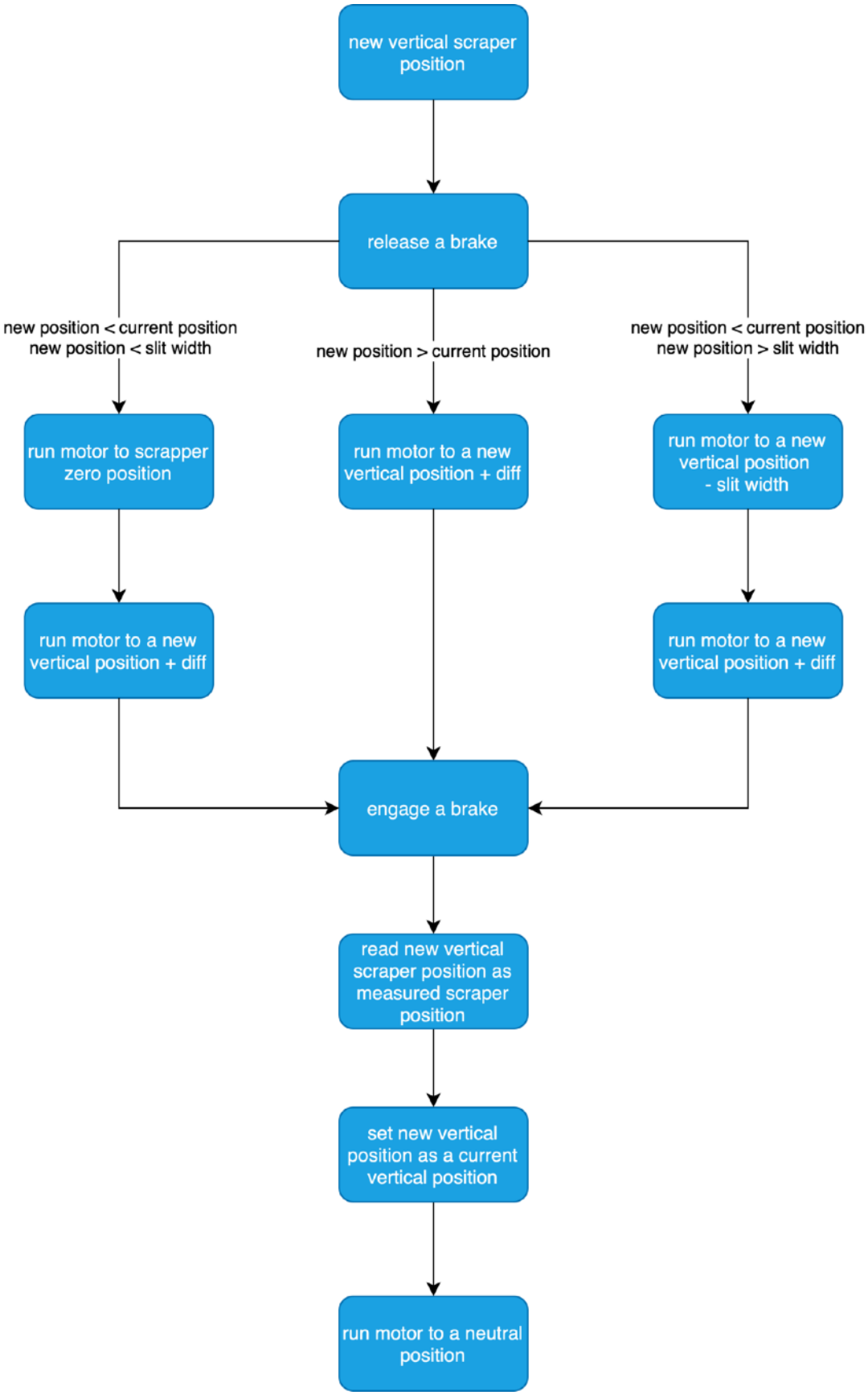


Figure 5.12.: Diagram of vertical scraper control

5.3.4 Horizontal scraper

The control of the horizontal scraper can be divided into two cases, according to the relationship between the new and the current scraper position. Since the scraper moves horizontally, there is no risk of the scraper falling or making any type of self displacement.

Both cases start by releasing the brake. Both then continue by adjusting the scraper to the new position in one step. Engaging the brake, reading the motor's position, setting the new position as the current one and setting the motor to its neutral position. The only difference between these two cases is, that the *diff* constant must be added to the new horizontal scraper's position in the first case. The first case is, when the the new position value is higher than the current one. The *diff* constant is explained better in 5.3.1.

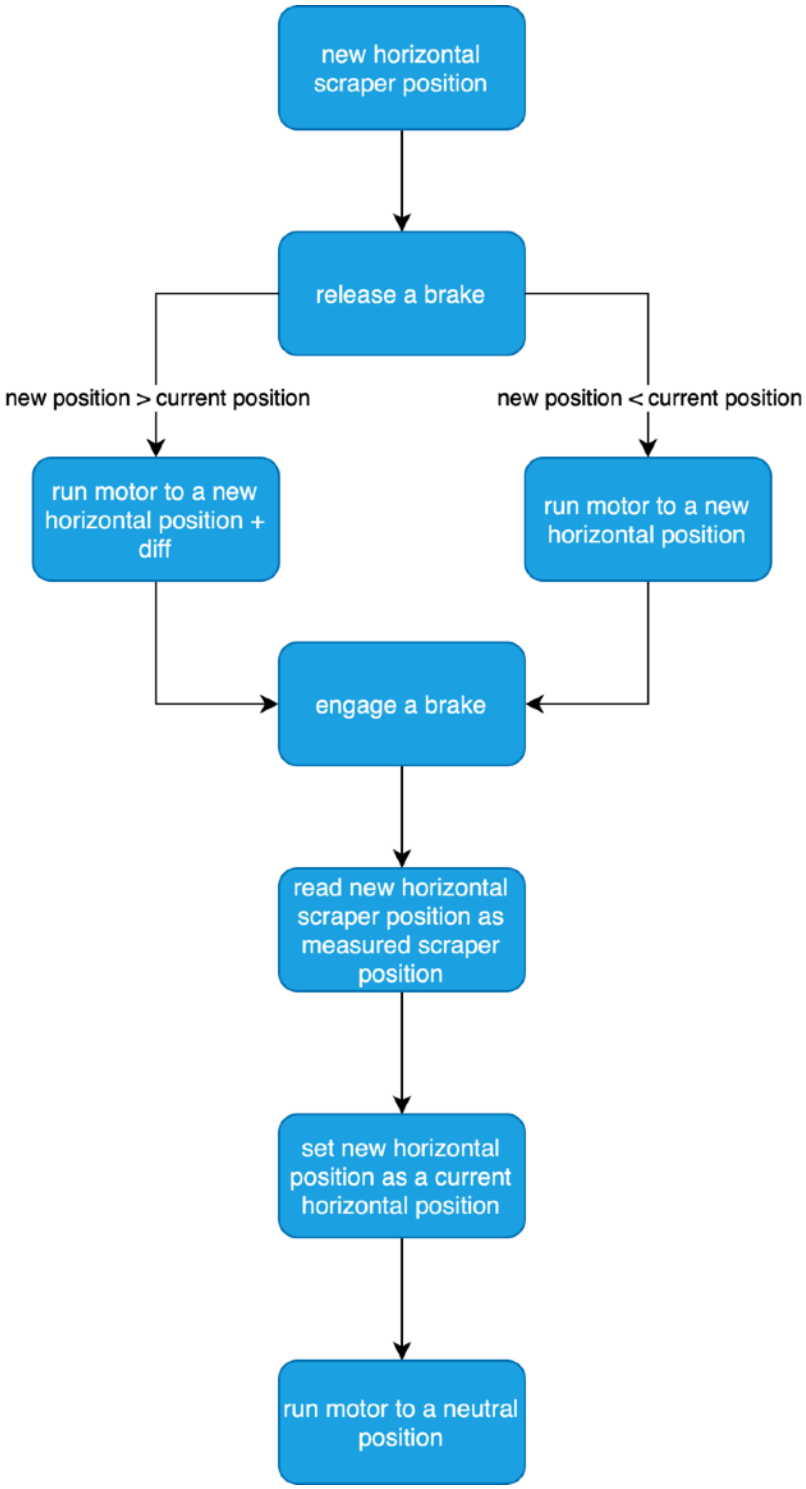


Figure 5.13.: Diagram of horizontal scraper control

5.3.5 OPC UA Server

For the control script, an OPC UA Server needed to be created within the script to secure the communication with the existing OPC UA Client. The structure of the Server is created according to the Client specifications so the new scrapers' control system would work without the need to change something in the Client.

Variables of the OPC UA Server:

- #*_scraper_position*: Float(set new scraper position)
- # *_scraper_measured_position*: Float(get scraper measured position)
- #*_scraper_motor_position*: Float(get motor position)
- # *_scraper_calibrated*: Boolean(True if calibrated)
- *air_flow_value*: Boolean(True if on)

The symbol # in variables means that those variables are for both scrapers. #*_scraper_position* variable is a writable variable and by using an OPC UA Client this variable can be set. #*_scraper_measured_position* and #*_scraper_motor_position* are variables that get their values by reading from the serial line and present these values to the Client. # *_scraper_calibrated* variable informs the Client whether the calibration was proceeded by Boolean True or False. The calibration can be called by a Client at any time by setting scrapers' positions to 0. Last variable is a writable variable *air_flow_value* by which a Client can control the air-flow system.

5.4. Control application

The control application was created mainly for testing purposes of the scrapers' control script. The control application consists of a graphic interface and a server with an OPC UA Client. The graphic interface was made as a web page using html programming language. A server with an OPC UA Client establishes the communication with the scrapers' control script.

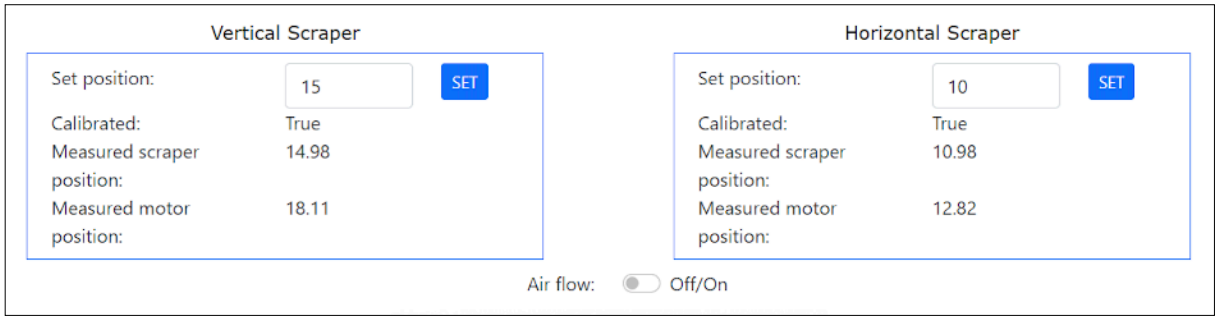


Figure 5.13.: Control application interface

The application allows to set both scrapers' positions and displays the current scrapers' positions after each repositioning. This feature allows the user to observe whether the scrapers have reached the required position and to estimate the deviation of the current scraper position from the required one. The application also shows a current motor position in real-time and enables to engage and disengage the air-flow system. The information on whether the scrapers are calibrated is represented by a simple Boolean True or False.

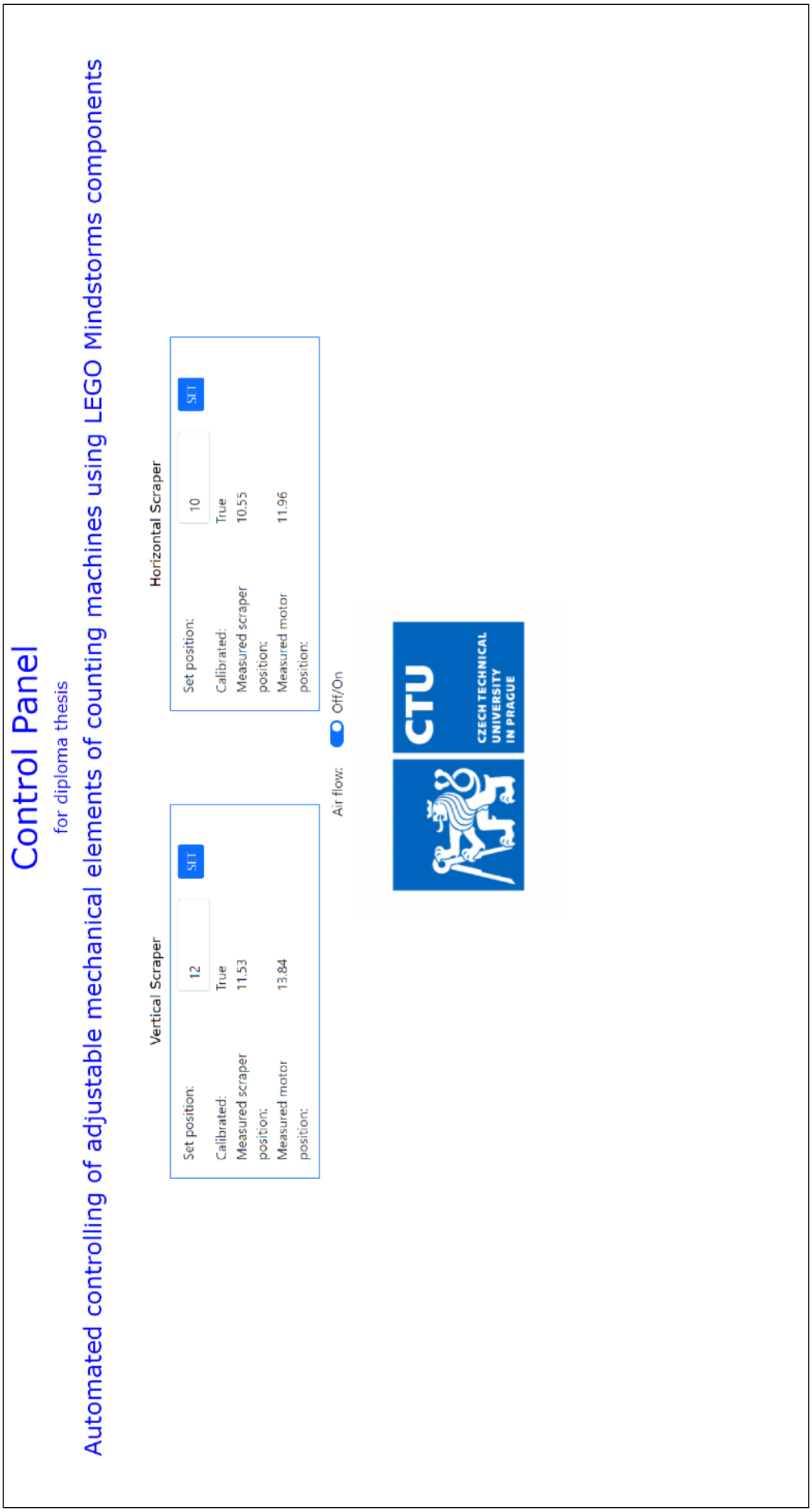


Figure 5.14.: Control application

5.4.1 Testing of the scrapers' control system

The testing of the scrapers' control system was successful in a way of proper function of every component of the system. However, the accuracy of the scrapers' positioning was often insufficient as shown in figure 5.15. The maximum deviation limit should ideally be below 0,5 millimeters.

NUMBER OF MEASUREMENT	VERTICAL SCRAPER POSITION		HORIZONTAL SCRAPER POSITION	
	SET	MEASURED	SET	MEASURED
1	25	23.2	20	21.28
2	15	14.98	10	10.98
3	30	27.29	21	20.87
4	20.5	19.6	10	10.55
5	4.8	4.49	25	24.47
6	30	27.68	28	29.13

Figure 5.15.: Table of measurements

Conclusion

Both main goals of the thesis were successfully fulfilled. The Stepper motor was replaced by a LEGO motor and the control script was designed and tested. Even though the new scrapers' control system work efficiently, the accuracy of its positioning is not sufficient.

When the LEGO motors were tested without load, their accuracy was around one or two degrees which would be enough for the purpose of adjusting the scrapers on the counting machines. However, the testing with load using the controlling application shows unreliability in LEGO motors' accuracy. In most measurements with load, the accuracy of the LEGO motors was not sufficient. The reason for this unreliability is most likely the build and the control system of the LEGO motors.

Another problem with LEGO motors is, that they need to be controlled from a LEGO Hub. The serial line, which is used for connection between the LEGO Hub and the control script that runs on Raspberry Pi, is not an optimal way for this type of data exchange. The serial line is easily overwhelmed, causes relatively large time delays in this use and all the information from the motors must be obtained as strings by reading the serial line in loops. By obtaining the motors' information directly to the control script, the script could be more efficient and with less errors.

In conclusion, in this particular case, the servomotors are efficient substitutes for the stepper motors. They eliminate the requirement of human interaction to the minimum. Calibration can be automated and is much quicker, and the scrapers' adjustment is more precise. However, it is suggested to swap the crank for a gearbox, which would reduce the force that the motor must exert. Due to the fact that there is a large number of counting machines, this solution's main disadvantage is its price.

This diploma thesis also proved the OPC UA technology to be very effective technology as very efficient in the integrating of new components to the existing systems.

Bibliography

[1] *Pre-pack bags - LEGO® History - Lego.com CZ*. (n.d.). Lego.com. Retrieved May 16, 2022, from https://www.lego.com/cs-cz/history/articles/f-pre-pack-bags

[2] *LEGO Digital Prepack - Research Report 2019*. (n.d.). Cvut.cz. Retrieved June 16, 2022, from https://www.fa.cvut.cz/cs/vyzkum-a-spoluprace/publikace/12017-lego-digital-prepack-research-report-2019

[3] Nováková, A. (2021, April 21). The LEGO Group and CIIRC CTU increase the efficiency of the production and replace routine work with AI. Cvut.Cz. https://www.ciirc.cvut.cz/lego-ai/

[4] Rinaldi, J. S. (2016). Opc ua - unified architecture: The everyman's guide to the most important information technology in industrial automation. Createspace Independent Publishing Platform.

[5] Wikipedia contributors. (2022, July 20). *OPC Unified Architecture*. Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=OPC_Unified_Architecture&oldid=1099355577

[6] *Opc ua*. (n.d.). Paessler.com. Retrieved August 1, 2022, from https://www.paessler.com/it-explained/opc-ua

[7] Krause, B. (2022, May 6). *What is OPC UA? A practical introduction*. System Integration with the OPC Router; OPC-Router.com. https://www.opc-router.com/what-is-opc-ua/

[8] Basler, A. G. (2021, April 27). OPC UA: Communication protocol for Industry 4.0 and IIoT | Basler.

[9] *UA ANSI C Server Professional: OPC UA Subscription Concept*. (n.d.). Unified-automation.com. Retrieved May 12, 2022, from https://documentation.unified-automation.com/uasdkc/1.4.0/html/L2UaSubscription.html

[10] Frey, J. (2019, October 16). *Erste Schritte zu einer gemeinsamen Sprache der Holzverarbeitungsindustrie*. AIT GmbH & Co. KG. https://www.aitgmbh.de/blog/news/erste-schritte-zu-einer-gemeinsamen-sprache-der-holzverarbeitungsindustrie/

[11] *Stepper Motor*. (2015, July 16). Techopedia.com; Techopedia. https://www.techopedia.com/definition/13345/stepper-motor

[12] (N.d.). Omc-stepperonline.com. Retrieved June 1, 2022, from https://www.omc-stepperonline.com/stepper-motor-driver

[13] Wikipedia contributors. (2022a, June 12). *Stepper motor*. Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Stepper_motor&oldid=1092819045

[14] Wikipedia contributors. (2022a, January 17). *Servomotor*. Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Servomotor&oldid=1066310615

[15] Earl, B. (2014, May 5). *All about stepper motors*. Adafruit Learning System; Adafruit. https://learn.adafruit.com/all-about-stepper-motors

[16] (N.d.). Electrical4u.com. Retrieved June 1, 2022, from https://www.electrical4u.com/what-is-servo-motor/

[17] Gastreich, W. (2018, August 27). *What is a Servo Motor?' and How it Works*. PLC Programming Courses for Beginners | RealPars. https://realpars.com/servo-motor/

[18] Wikipedia contributors. (2021, November 2). *Lego Mindstorms*. Wikipedia, The Free Encyclopedia. https://cs.wikipedia.org/w/index.php?title=Lego_Mindstorms&oldid=20603903

[19] LEGO System A/S, LEGO® MINDSTORMS® Inventor, *Robot Inventor Companion App*, version 10.4.0, 2020

[20] *LEGO Black Technic Pin with Friction Ridges and Slots (2780 / 61332)*. (n.d.). Brickowl.com. Retrieved August 10, 2022, from https://www.brickowl.com/catalog/lego-black-technic-pin-with-friction-ridges-and-slots-2780-61332

[21] *Lego System: Introduction*. (n.d.). Tum.de. Retrieved April š, 2022, from https://www5.in.tum.de/lehre/praktika/grapra/WS00/grapra01/Lego/LegoIntroduction.html

[22] Wikipedia contributors. (2022, May 11). *Solenoid valve*. Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Solenoid_valve&oldid=1087207857

[23] Hohmann, J. (2016, March 21). *Pneumatic circuit symbols explained*. Library.automationdirect.com. https://library.automationdirect.com/pneumatic-circuit-symbols-explained/

[24] Festo. (n.d.-a). *Air solenoid valve*. Festo.com. Retrieved May 21, 2022, from https://www.festo.com/us/en/a/8042539/

[25] Festo. (n.d.). *Solenoid valve*. Festo.com. Retrieved May 21, 2022, from https://www.festo.com/tw/en/a/566455/

[26] *Automation HAT*. (n.d.). Pimoroni.com. Retrieved August 10, 2022, from https://shop.pimoroni.com/products/automation-hat?variant=30712316554

[27] *Hub Documentation — hub-api documentation*. (n.d.). Github.Io. Retrieved April 10, 2022, from https://lego.github.io/MINDSTORMS-Robot-Inventor-hub-API/

Appendix

scrapers_control.py