https://doi.org/10.7494/miag.2021.1.545.25

PAWEŁ KARELUS
KRYSTIAN SZOPA

Design and construction of a three-axis Pick and Place manipulator

The article presents the design and implementation of a three-axis Pick and Place manipulator based on OpenPnP software. It is designed for the placement of SMD components on PCB boards. A ready-made frame was used for the project, which required the selection of appropriate motors and the design of the required components for the Z-axis, which were design in Solidworks software. A belt drive was used to construct the manipulator as well as pneumatic components for lifting the SMD parts. Appropriate cameras were also selected and configured along with lighting, allowing the software to automatically check the progress of the machine. The next step was to select the electronic components of the machine and configure them before OpenPnp software was configured and prepared for the automatic placement of SMD components.

Key words: *manipulator, pick and place, SMD, OpenPnP*

1. INTRODUCTION

Pick and Place manipulators [1], as the name suggests, should perform the activities of picking up and moving components from one place to another. There are many solutions for such designs, and the two main solutions are the robotic hand (Fig. 1) and the two- or three-axis assembly manipulator (Fig. 2), which is described in this paper.

Fig. 1. Robotic hand style manipulator designed by AGH Space Systems

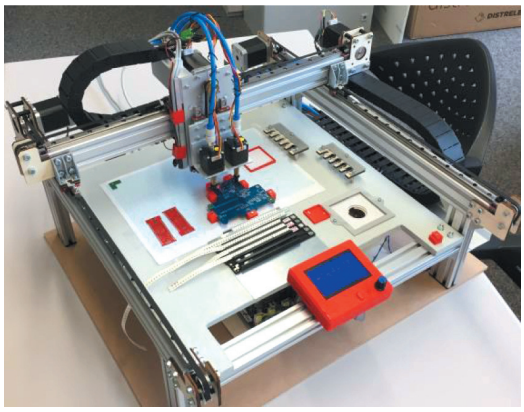

Fig. 2. A simple three-axis assembly manipulator

Its task will be to pick up and place SMD elements (surface-mount device. This abbreviation stands for all electronic components which are placed on the surface of PCB boards, i.e. resistors, capacitors or processors) on printed circuit boards. In plants engaged in mass production of complex electronic components, such manipulators do a large part of the work. However, they are industrial, which means that they have a relatively large size, as well as being very expensive to purchase and maintain.

Small factories with a small number of projects can outsource the assembly of the board, but this is also an expensive solution, and laying out the elements by hand often takes many hours and is tiring work due to its high repetition and small size of the elements. In order to reduce the time needed for the assembly of elements, it is possible to use a Pick and Place manipulator, which can be purchased or made by yourself. This frees up the worker to perform more complex tasks, while the machine performs simple, repetitive and long-lasting work.

There are many commercial manufacturers of Pick and Place manipulators. There are designs available in various sizes and throughputs, from smaller designs having single heads, to larger industrial designs having dozens of heads [3–4]. One example of such a machine available on the market is the LitePlacer manipulator (Fig. 3) manufactured by a company of the same name from Finland.

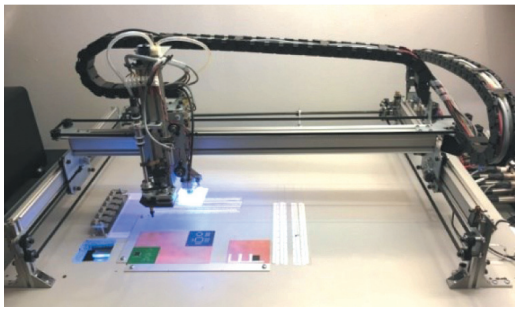

Fig. 3. LitePlacer manipulator [3]

Industrial designs that have very high speed and can stack tens of thousands of components per hour and have hundreds of inputs for various SMD components, which is needed if the board designs are complex are also available. However, they are used for mass production and are only applicable in large plants. One of the available solutions is the JUKI RS-1R machine (Fig. 4), manufactured by the Chinese company ETA. Such a manipulator is used as part of a production line, it has automatic inputs and outputs in the form of belt conveyors, thanks to which it is possible to connect several machines performing successive points in the production process. It has a throughput capacity of about 47,000 parts per hour, an impressive number.

However, if the costs associated with the purchase of a manipulator are too great, it is possible to design one's own device using the available open source software. On the other hand, this is a time-consuming solution and requires knowledge of mechanics and mechanical engineering, as well as electronics and programming basics.

Fig. 4. JUKI-RS-1 manipulator [4]

2. HISTORY AND CHARACTERISTICS OF PNP MANIPULATORS

The history of Pick and Place manipulators dates back to the 1930s, but it was not until the post-World War II era that the first industrial units were created. With the development of technology and economics, the demand for the number of products produced by factories increased significantly, this increased the interest in machines that will perform simple and repetitive tasks instead of humans, especially in environments that threaten the health or life of workers [5].

The algorithm of assembly manipulators is not complicated at first glance. It is enough to perform only a few basic steps, which need to be repeated many times. In practice, calibration of each step takes time because of the required precision and repeatability.

The Pick and Place manipulator control algorithm requires several steps (Fig. 5). It starts with a CAD design of the PCB including the components to be laid out in it. This design is then loaded by the software installed on the computer that serves as the main controller of the device. This software matches the individual elements on the board to the elements on the feeders and creates instructions in a language understood by the electronic controller, most often G-code. The controller receives the instructions and executes them by sending electrical pulses to the appropriate connectors, which causes the machine to move in the proper directions to achieve the intended effect. To detect the correct positioning of the picked components, cameras are most often used, from which the image is processed by the software, which will be described later in this article.

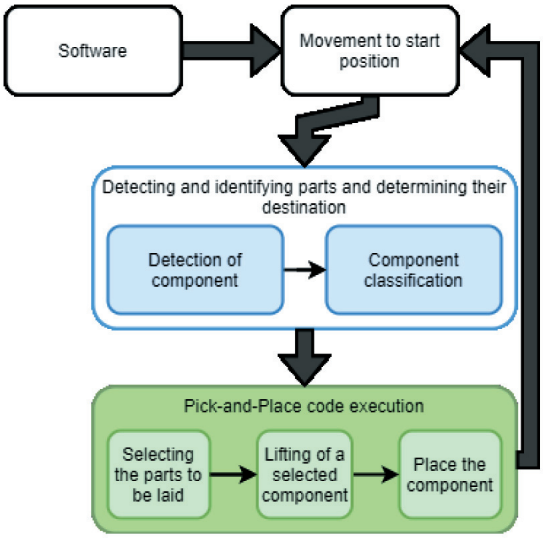

Fig. 5. Schematic of the PnP manipulator [6]

3. DESCRIPTION OF THE COMPLETED STRUCTURE

3.1. Project preparation

The beginning of any project is made up of assumptions and objectives. In this case, a previously prepared frame will be used (Fig. 6). This reduced the cost needed to construct the manipulator as well as the time. This design already had the X and Y axes ready with the screw drive applied. The work steps adopted were as follows:

- selecting and installing the motors that drive the axles,
- selection of the ratio of belt transmissions used to drive the X and Y axes,
- design and construction of Z-axis with mounts for two heads,
- design holders for sensors,
- mounting cameras,
- wiring of the machine,
- mounting the manipulator to a table with wheels,
- assembly of trays with SMD components,
- installation of power supply unit, controllers, keypad controller and emergency stop switch,
- connection and configuration of the controller using software designed for 3D printers,
- OpenPnP software configuration.

At the beginning, the mechanical structure of the manipulator will be made, then the electronic components will be connected and the final stage will be the installation and configuration of the software. The use of two heads is motivated by the fact that with the planned use of a drive based on belt transmission. The control of them runs in a simple way. Mounting them on two sides of one pulley means that when one head is moved in one direction, the other one moves in the opposite direction. The use of two heads also means that the manipulator works faster than if only one head were used. Figure 6 gives an overview of the frame structure.

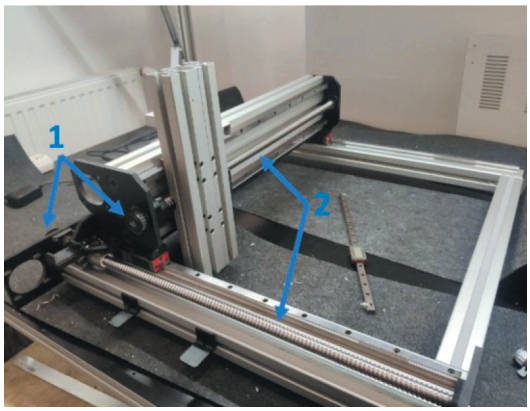

Fig. 6. Initial appearance of the frame: 1 – screw drives; 2 – pulleys

3.2. Mechanical design

Due to the fact that an off-the-shelf frame was used to construct the manipulator, some components were chosen in advance. These elements include the screw drive in the X and Y axes direction, or torque transmission from the motor, which is done through a belt transmission (both elements can be seen in Figure 6). As these drives will not transmit large torques and the highest possible speed is required, a gear ratio of 1 was selected. Nema 24-60HS88 series stepper motors with a nominal torque of 3 Nm were selected. The pulleys were selected, taking into account those already mounted on the propellers, to obtain the assumed ratio. The only possible way turned out to be the selection of an identical version of the wheels, namely HTD 22-5M-09 pulleys having 22 teeth and a pitch diameter of 35 mm. The axles are mounted to linear guides on specially designed carriages in order to achieve the lowest possible resistance to motion and stability of the structure.

3.3. Z-axis design

The Z-axis design (Fig. 7) was made in Solidworks according to the assumptions presented earlier. This

required a motor to be mounted on the axis along with a belt transmission to drive the heads, which were mounted in specially printed holders on linear guides and connected directly to the belt transmission. This caused the heads to move alternately. There is also a pneumatic system on the axle, causing a vacuum in the head, enabling the lifting of SMD elements.

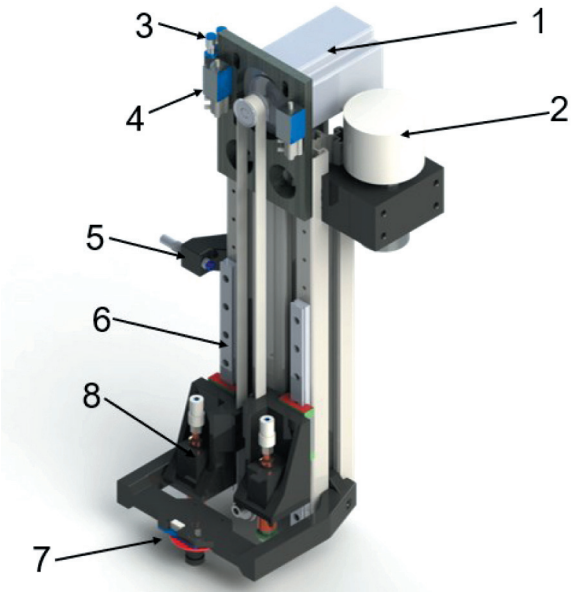

Fig. 7. Design of drive system in Z-axis direction made in Solidworks: 1 – stepper motor; 2 – pump; 3 – pneumatic distributor; 4 – solenoid valve; 5 – inductive limit sensor; 6 – linear guide; 7 – top camera; 8 – head

This system consists of the following elements: a vacuum pump, a distributor, two pneumatic solenoid valves with one input and two outputs and pneumatic hoses. The purpose of the pump is to create vacuum in the hoses enabling lifting SMD elements. The head is made of a hollow shaft, allowing simultaneously for rotation of the tip with the element lifted by means of previously mentioned vacuum. For proper functioning of the software there is a need for a camera, which will be in a fixed XY position in relation to the heads, it allows to detect elements on the tapes as well as the position of the PCBs.

Both the motor and the pneumatic system were mounted on top of the aluminium profile. The motor was mounted in such a way that it is possible to fix a gear on its pinion, and the motor mount itself allows belt tension. Solenoid valves were used because one pump was attached, if the elements are to be placed one after the other, turning off the pump would result in the loss of vacuum generated in both heads, which would cause both elements to drop. By switching the valve instead of turning off the pump, the pressure can be equalized in one head only. The manifold allows two valves to be connected to one inflator. Pneumatic lines were not routed in the CAD model due to the fact that they are connected loosely, so it is difficult to predict their routing.

The inductive limit sensor (position 5, Fig. 7) is fixed so that its position can be adjusted at any time. The fastening to the aluminium profile is carried out by means of a screw with lock nut, which, together with a specially mounted bracket, results in a rigid position when the fastening screw is tightened. The sensor itself has a thread on almost its entire length, which also gives the possibility of adjusting the appropriate position.

The heads (item 8 on Fig. 7), responsible for the operation of lifting and putting down the elements, are mounted in specially designed holders and screwed to the carriages on linear guides. Owing to such a construction, their movement is smooth and without any problems connected to friction. The movement itself takes place because of the connection of the holder to the toothed belt by a specially designed fastening. This combination of elements limits the movement of the gear to a small range, but the required head movement is short.

The upper camera (position 7, Fig. 7) should be at a fixed distance from the heads, so it too was fixed using a suitably designed bracket.

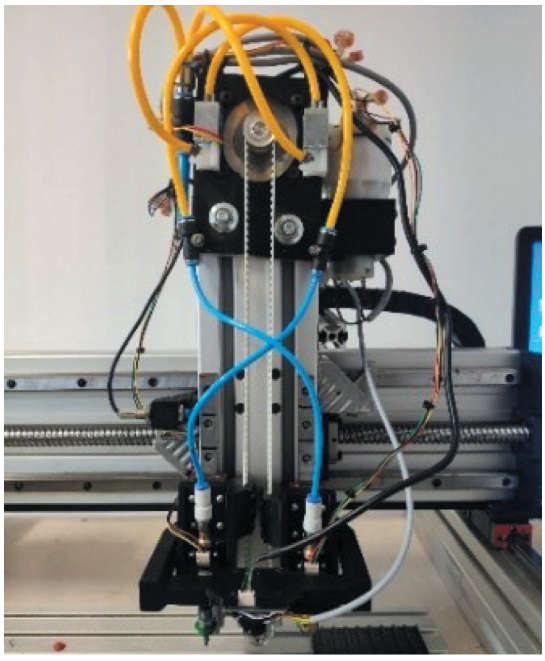

Fig. 8. Final appearance of the drive system in the Z-axis direction including the lifting system

The final appearance of the Z-axis structure is shown in Figure 8. The wires themselves are cross-connected, i.e. the left valve is connected to the right joint and the right one is connected to the left head. This solution results in more freedom of wire movement during operation and minimizes sharp wire bends. SMD elements in most cases do not have a large mass, but to make sure that any larger microcircuit, such as a processor was lifted properly, we had to take care of the lowest possible pressure value at the end of the head.

3.4. Control algorithm

The control of the machine movements is either automatic or manual. The manual way is provided for configuration, each step can be done on your own by running individual options in the OpenPnp software, but the default way is automatic control. To make it possible, it is first required to enter all the necessary data into the program and the controller. The control algorithm (Fig. 9) is shown for automatic control only. The movements of the machine take place on the basis of instructions sent in the G-code language. The instructions are created by the OpenPnP program sent to the Arduino microcontroller, which converts the code and sends signals to the appropriate machine components, causing them to move. The code is sent on the fly, so a connection between the controller and the computer is required throughout the machine's operation.

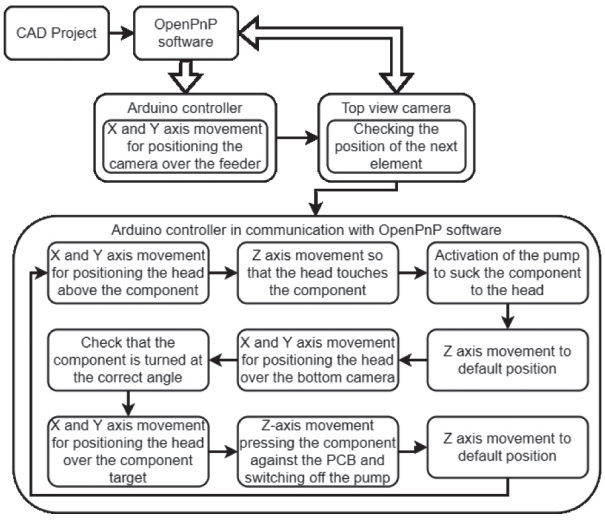

Fig. 9. Algorithm for manipulator control

However, not all elements are connected directly to the controller. The top and bottom cameras are connected via a USB port to the computer that operates the machine, so that the OpenPnP software has access to the image and can appropriately select the parameters of distance or rotation by which to move the head. The second element are LED lights mounted next to the cameras. They are used to illuminate plates and tapes with SMD elements in the case of the top camera and elements sucked to the head in the case of the bottom camera. They are connected to Arduino Nano controller, which is then connected to appropriate pins on the main controller. Controlling the lights will be described later.

4. ELECTRONIC COMPONENTS AND THEIR SOFTWARE

For the construction of simple PnP manipulators, the most common components used are those created to support 3D printers, they are readily available, not expensive and relatively easy to configure to support the manipulator. One of the most important practical requirements is that the controller software is able to support the original two extruders in the printer. However, if this is not the case, then the configuration code can be changed to support the manipulator but this is certainly problematic. Taking this into consideration, the MEGA 2560 microcontroller was used along with the RAMPS 1.4 overlay. This overlay is an extension of the MEGA 2560 microcontroller, it allows the connection of stepper motors along with stepper motor drivers. A simple and inexpensive A4988 stepper motor driver was chosen, two of which are ultimately needed, one driver supports only one motor, however they are not capable of controlling X, Y and Z axis motors. Their job is to operate the axis of rotation. One of these controllers is fitted with a view to future expansion of the machine. A HY-DIV268N-5A two-phase controller was chosen to control the motors of the main three axes. It has the ability to simply change the current as well as the number of micro steps. This configuration allows selected stepper motors to be controlled without any problems. The RAMPS also gives the possibility to connect limit sensors, which were also used in this project. Using the remaining pins, you can control additional elements, in the case of the manipulator these are solenoid valves, a pump or LED lights, for which you still need an Arduino Nano, because the signal coming out of the microcontroller is a PWM

signal (Pulse-Width Modulation [7]). The microcontroller sends values starting from 0 all the way up to 255, so it is a digital signal that is not able to power the lights. So the signal is sent to the Arduino Nano microcontroller, which when it receives a value of 0 turns on the lights and when it receives a higher value turns them off. In practice, these three elements, not including Arduino Nano, form a whole. There is no need to configure each element separately, 3D printer driver software is uploaded to main microcontroller i.e. Mega 2560, which controls the rest without the need for external configuration. By adding an additional microcontroller to this configuration, we get the possibility of extending the manipulator with additional elements, in the case of this machine, these are LED lights, but in the future, you can extend the design by connecting to it also, for example, automatic feeders of SMD elements.

4.1. Configuration of the MEGA microcontroller

The most important changes to the default microcontroller configuration were:

- selection of the communication baud rate between the controller and the computer,
- setting of minimum and maximum position value of individual axes,
- setting of limit sensors (normally open, X and Y axis minimum sensor, Z axis maximum sensor),
- setting the resolution of stepper motors and their maximum speed and acceleration.

Due to the fact that the software used is intended for 3D printers, only three axes were provided in it. To be able to use the fourth axis, which acts as a rotation of the head, it was connected to the extruder connector, which required disabling support for temperature sensors used in printers.

After setting all the options, the finished program was installed on the controller. The next step was to configure Arduino Nano microcontroller.

4.2. Configuration of Arduino Nano microcontroller

LED boards are attached to the camera mounts. Their function is to illuminate the manipulator when the software downloads the image. Since they are mounted parallel to each other but in opposite directions, they should not be lit up at all times when the machine is running. Light, illuminating the image from the lower camera, may result in interference with the image from the upper camera and vice versa. For this reason, the configuration of the microcontroller is not enough to get the lights working properly, but it is the first step. There are cables connected to pins 7 and 8 of the device that carry the enable signal. By using the appropriate code, the lights can be controlled to turn on and off by applying low and high voltage. The LED boards are connected in series, with 16 diodes on each one. The use of simple conditional instructions allows you to turn on the entire board virtually simultaneously, if the voltage at the output changes to low. The colour is given in RGB format, the value 250, 0, 0 is given, which means that the lights glow red. This shade was chosen because no part of the manipulator's structure is in the same colour and it causes noticeably less light reflection than when using white light.

5. OPENPNP SOFTWARE CONFIGURATION

Ensuring the proper operation of the manipulator also requires correctly configured software to control it. Firstly it was checked that all axes and sensors were working correctly. The OpenPnP software requires the machine coordinate system shown in Figure 10. Once satisfied that the components of the design were working correctly, moved on to determine the steps per mm for the stepper motors.

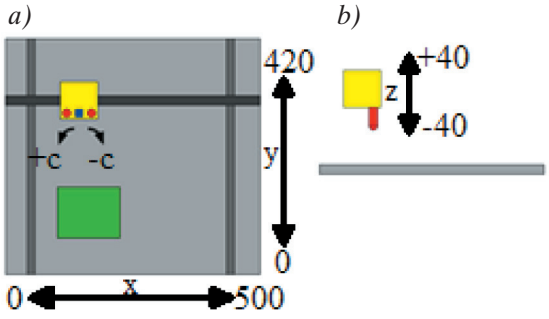

Fig. 10. Coordinate system of the manipulator: a) top view; b) side view [8]

The resolution of stepper motors can be very easily calculated by dividing the number 360 by the step angle of the motor [9], which is 1.8°. The quantity calculated in this way determines the number of steps needed for a complete rotation of the motor. How-

ever, to calculate the value of steps per mm, the formula for screw drives should be used:

$$k_r = \frac{l_k \cdot m_k}{P} \tag{1}$$

where:

l_k – number of motor steps per full revolution,
m_k – number of micro steps,
P – pitch of the propeller thread.

The assumption is 16 micro steps. For the formula we also need to know the pitch of the propeller thread, which is 8 mm. Knowing all these values, we can substitute them into equation (1) obtaining:

$$k_r = \frac{200 \cdot 16}{8} = 400 \tag{2}$$

It follows that it takes 400 steps for the motor to move an axis by 1 mm. To be sure that the value has been calculated correctly, it must be checked in practice.

To calibrate the basic settings of the OpenPnP software, the manual [10] created by the developers and available on Github was used.

5.1. Initial software settings

The first step is to install the OpenPnP software. Start by selecting a software driver so it can properly communicate with the microcontroller. For all designs where the control is based on the G-code language, there is a "GcodeDriver" option. Next, you need to select the port to which it is connected, and a baud rate of 115,200 Bd.

5.2. Calibration of the top camera

The first element to be calibrated is the top camera, which is mounted on the Z-axis. For cameras connected via USB ports it is recommended to use OpenPnp CaptureCamera driver and this is what was chosen. This option is the code that allows the software to correctly read the camera image and control it. The image format has also been set. The parameters of camera image processing such as brightness, saturation and white balance have been set so that the image looks the same at any time of day. This allows for correct operation of algorithms processing the image, e.g. to detect the SMD component on the tape, regardless of whether the manipulator is in a lit room or not.

The top camera, for best illustration, is called CameraTop One of the most important settings is the calibration of units (in this case millimetres) per image pixel. If this parameter is measured incorrectly, it can result in erroneous machine operation, because at some points in the operation algorithm the actual distance is measured from the image. The calculated values are 0.0662 mm per image pixel. The top camera is the zero point for the software. The next settings are the transformations of the images coming from the cameras. According to the recommendations, the image from the top camera should look like what a person looking at the manipulator from above sees, i.e. there should be elements in the top of the image that are further out in the Y-axis, etc. If the fixed camera does not provide such an image, it can be inverted vertically, horizontally or rotated by a given angle. It is also possible to crop the image, a useful operation as the camera lens resembles a hemisphere. As a result, the image does not have constant dimensions (the number of pixels per 1 mm around the centre of the image is not equal to the number of pixels per 1 mm at the edges of the image).

5.3. Calibration of the bottom camera

The configuration of the bottom camera is similar to the configuration of the top camera, however the set values differ because this element is set in a different place and is responsible for detecting the rotation of already lifted parts and the position of the head tip. The camera was named BottomCamera to maintain the naming theme. The mm value per image pixel equal to 0.017 mm was also calculated. The image transformation settings were also chosen so that the image is similar under all external lighting conditions. One of the most important settings for the bottom camera is to determine its position, but it must be a position where the tip of the head is perfectly in the centre of the vision. This cannot be calculated, therefore, without first calculating the distance at which the head is located from the upper camera, which is the zero point for the software.

5.4. Nozzle calibration

Initially, only one head was configured so that it would be easier to verify the correctness of the design. It was given the name N1. Then, to enable

automatic operation of the pneumatic components, appropriate switches were added, which send G-code commands to turn on or off the relevant components. Thus, a Vacuum switch was created, which causes the pump to turn on or off, and two switches N1_VAC and N2_VAC (denoting the solenoid valves for head one and head two). The next configuration step was to determine how far the head was from the manipulator zero point. Performing this step also allowed us to determine the location of the bottom camera. Determining the position of the camera also allowed us to determine the misalignment error of the machine, which, if too large, can affect the incorrect performance of the manipulator.

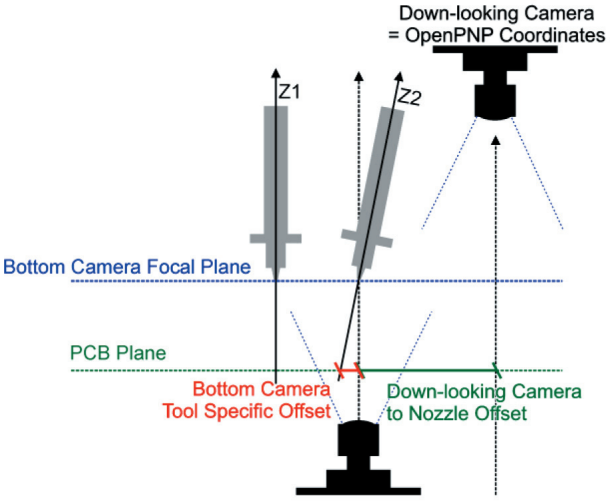

Fig. 11. Visualization of misalignment error [11]

As shown in Figure 11, if the head is not mounted in an axis perpendicular to the plane of the PCBs, this results in offset. Additionally, if the nozzle itself is not made with concentricity or the head tip is mounted in a way that causes misalignment, it is difficult to determine the point where the head tip touches the PCB, due to the fact that the offset changes direction and/or value with the rotation of the head motor. Thus, an algorithm that processes the camera image and detects the head tip is used for this process. Once the bottom position of the camera is determined, the head tip should be in the centre, then the algorithm performs a rotation of the head motor so that the sum of these rotations is 360°. This gives an idea of how the position of the head tip changes in terms of rotation at each location. After each individual rotation, a camera view is taken and processed by specially selected algorithms to detect where the head tip is located. It consists of several smaller stages, which ultimately should lead to the detection of a single circle at the location. This, along with the previously calculated number of mm per pixel of the downstream camera, allows the algorithm to calculate how much misalignment error there is at a given motor rotation. This result is stored and then added or subtracted when performing automatic moves. However, it should also be taken into account that no limit sensor is fixed in the axis of rotation of the head. This causes the rotation position to be set to 0 each time the machine is turned on, so that added values may not work correctly and increase the error instead of eliminating it. It is therefore important that this calibration is done after every basing of the manipulator.

5.5. Basing the manipulator

When the machine is powered up, the axis position is automatically set to 0. So in order for the software to know the exact position of the machine, a basing operation must be performed. This consists of three steps:

1. access by axes to the end points according to the manually entered G-code,
2. drive the camera over a special point located on a table with a fixed position to ensure as much baseline repeatability as possible,
3. drive the head over the lower camera to calibrate the misalignment error (described in the previous section).

The first stage consists of several commands in G-code language. They ensure safe arrival of each axis to the limit sensors. When basing automatically, it should be remembered that there may be a situation when the head is in a low position.

SMD components often have small dimensions, so every option available to increase precision should be used. One of those options includes moving the camera over a specially prepared black point located on a table with a fixed position in relation to the structure. Its position was included in the configuration file, so after the first stage of basing and moving over this point one of the algorithms processing the image of the camera, checks if it is located perfectly under the camera. If there is a shift, the position of the machine is corrected. If the point is not detected, the basing operation is interrupted and it has to be performed again. The next step is to drive the head over the lower camera to calculate the misalignment. However, if the misalignment is not calculated correctly, the basing operation is also interrupted.

6. IMAGE PROCESSING ALGORITHMS

In order for the manipulator system to work automatically, suitable algorithms are required to detect the required elements in the camera image. One algorithm consists of several smaller algorithms that can be selected according to the requirements, ultimately leading to the detection of the component. Of course, they differ depending on what component needs to be detected. The selection of appropriate algorithms should be done after the configuration of the image coming from the cameras. It is important that the algorithms work similarly in all lighting conditions.

However, the general principle of the image algorithms is similar. Each of them starts by retrieving a frame from the camera image after the settling time has elapsed. Then it reduces the downloaded frame in order to minimize the number of elements in the image. Subsequently, image processing operations are performed and the final result is drawing on the initially downloaded frame of the detected element – a rectangle or a circle. Nonetheless, there are many of the algorithms used by the camera software due to the fact that it is used to detect raised SMD elements. The algorithms used by the downstream camera can be divided into two main directions, as those used to detect lifted parts look very similar, mostly differing only in the selection of single values in the stages of the teardown process.

The second algorithm is the one for head tip detection. It consists of eight steps presented and described below. They should be selected in such a way that only one circle is detected which is exactly at the head end. If, for example, the detected circle is smaller or larger than the end of the head but its centre coincides with the actual centre of the component, this does not cause problems because the software only takes into account the centre point. The tip of the head is one, so there should be no more circles drawn in the image. The steps of the algorithm are as follows [9]:

1. ImageCapture – when the camera fix time expires, one frame is taken from the bottom camera image.
2. MaskCircle – algorithm bounds the previously taken frame with a circle of 350 pixels in diameter, the rest of the image is covered with black colour.
3. Bgr2Gray – performs colour conversion from the BGR system (24 bit colour value storage) to the grayscale system.
4. Threshold – this algorithm converts the image into two colours – black and white.
5. BlurMedian – performs pixel averaging. This removes noise from the image.
6. DetectCirclesHough – this algorithm is used to detect circles with a diameter within a specified range on the image.
7. ImageRecall – this algorithm overwrites the processed image with the initially downloaded frame.
8. DrawCircles – this algorithm draws the Circles detected in step 6 on the frame referenced in the previous step.

The algorithms used by the main camera differ from those used by the lower camera because they are mainly used to detect reference holes on the component tape or PCBs. During the automatic operation, the manipulator moves the upper camera over the position where the next component to be lifted should be and checks where and if the tape reference holes are located. In this way very precise calibration of long tapes (at which possible error was more noticeable) is not required, because with each lift the software checks and performs a possible correction of the position.

In theory, the algorithm for reference hole detection may resemble that for head tip detection. However, due to different camera settings and different lighting conditions for these elements, these algorithms had to be chosen differently. The last way of detecting elements is the algorithm for finding reference holes on PCBs. It is used to automatically determine the position of the PCB in the working field. It saves work for the user because the elements have to be placed with a very high accuracy so manually fixing the position every time one begins work would be tedious.

7. DESCRIPTION OF MANIPULATOR OPERATIONS

Once the manipulator configuration is done, it can be proceeded to perform the first automatic operation of the machine. When the machine is turned on, start by configuring the misalignment error. The operation of the algorithm detecting the head tip is shown in Figure 12. As can be seen in it, the image is first bounded by a circle and then its coding is changed to grey shades. In the next stage, pixels that are above

the selected grey scale intensity are detected. The image is cleaned from noise and in stage 6 a circle is found. Stage 8 shows that the algorithm has worked correctly. These instructions are executed for 6 different head rotations to detect the error with the current head axis configuration.

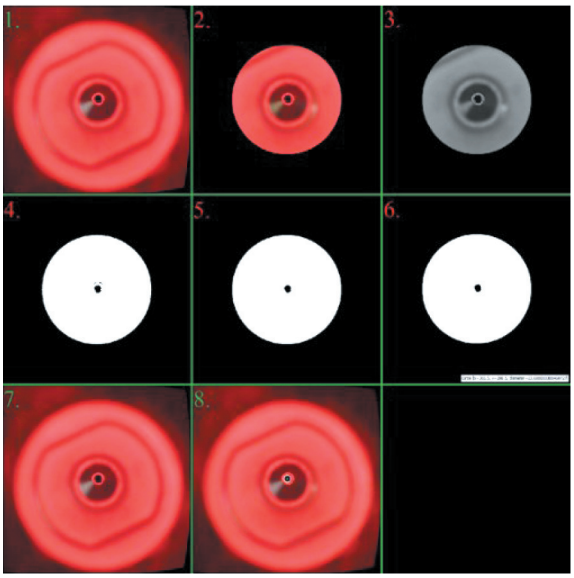

Fig. 12. Operation of the manipulator head tip detection algorithm. Numbering according to the algorithm presented in Section 6

The next step is to prepare a CAD file made in one of the many software designed for this purpose and save it in CSV or TXT format. Both of these formats contain a list of all the data needed to arrange the elements, such as position, rotation and type of element. There are many types of components that can be placed on the boards.

For the purpose of this article, we will describe the work of stacking one element, specifically the capacitor 0805 shown in Figure 13.

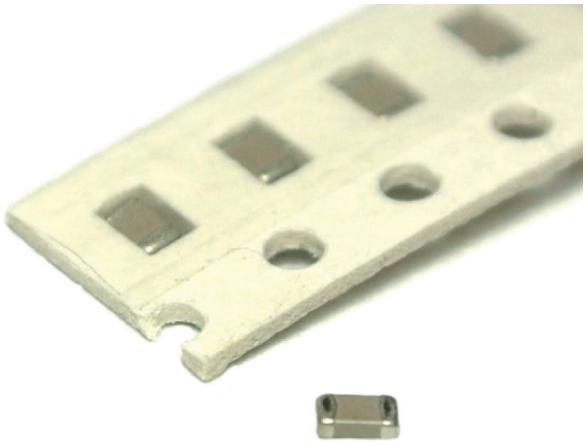

Fig. 13. Capacitor 0805 [12]

Next, the position of the PCB in the manipulator work area must be entered into the OpenPnP software, since the coordinates found in the files are counted from one of the corners of the PCB. There are usually three to four reference holes on PCBs. After checking the first, the manipulator then moves over the second and then the third (and then the fourth if there is one) and the whole process is repeated until the exact position and rotation of the PCB in the working field of the machine is calculated.

7.1. Automatic stacking

The next step is to switch on the automatic operation of the system. When the laying process begins, the robotic system starts by driving over the feeder with the elements. The software runs an algorithm to check the position of the reference holes of the strips, with the final result that all visible reference holes are found. Mounting several tapes next to each other is not a problem, because the instruction applied trims the image so that only the necessary fragment is visible.

It is also worth mentioning the lighting operation. LEDs should not be lit for the whole time the keypad is working, but only for a short moment when the image is taken from the camera. This is made possible by two applied scripts [13] written in JavaScript. The first one is run during the initialization of instructions related to reading the camera view. It checks which of the devices is to do this job and turns on the lighting intended for it, while turning off the LEDs located on the opposite side.

The operation of the second script is simpler than the previous one. Initially, a reference to the light switches is also written and then they are turned off. It is not necessary to use a conditional statement to check which lights were on, because turning off an unlit light does not cause any reaction. The entire process of retrieving the camera image takes less than a second.

After checking the position of the reference holes of the strip with elements, the software calculates a possible correction of the position if the circles found differ from what was stored in memory. Next, the X and Y axes are moved in such a way that the head is positioned exactly over the part to be picked. After its completion, the Z axis moves so that the tip of the head touches the element. At this point, the vacuum pump is activated and the part is lifted. Further, the manipulator moves the lifted component so

that it is in the centre of the lower camera image and at the same time at the height of the PCB on which it is to be placed. While performing this movement, it is also pre-rotated. Once the component is in the intended position, another image processing algorithm is run. In this case, the image is also bounded, this time by a circle, to reduce the number of elements that might impede the detection of the element.

The lighting used is red so that the element can be easily distinguished from the rest of the structure, which is done during stages three and four. Then the image is cleaned of noise and only a white imprint of the component on a black background remains. This ensures that the detected rotation and dimensions of the component are almost perfect. If the detected rotation angle of the component is different than the intended one, the head rotates by the calculated error. The algorithm does its job again if the correction is more than 5°. It also calculates the offset of the element from the centre of the image. The head does not always pick up the element exactly in the centre of its surface, which can cause small offsets during placement.

When all corrections have been calculated, the final sequence is executed. Z-axis rises to 0, X-axis and Y-axis move, positioning the capacitor exactly over the spot. The head then descends so that the component touches the plate and is pressed down. The end of the head is designed with a spring. This allows the component to be pressed against the PCB. When the movements are finished and the pressing time has passed (it was set at 1000 ms), the electro valve is switched on. The next movement is lifting the head to position 0. These movements are performed in parallel for both heads. When both elements are positioned, the algorithm starts from the beginning, i.e. the camera is positioned over the tape with elements, etc. When the last element has been positioned, the manipulator moves to the parking position and automatic operation is terminated.

7.2. Summary

The execution time depends on the complexity of the PCB design and the motor speed used. The more components to be placed, the longer the placement process will take. Increasing the speed will result in faster work with reduced accuracy. Also, be sure to provide enough components so that you don't run out of them while the job is being performed. If this happens, the algorithm will stop and the software will require user intervention. When the job is complete, the manipulator is in the parking position shown in Figure 14.

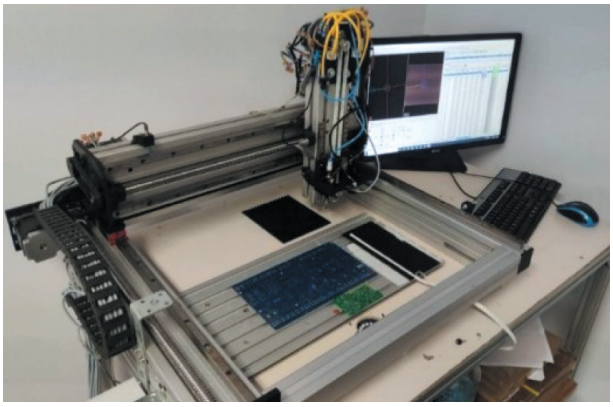

Fig. 14. Manipulator in the parking position

8. CONCLUSIONS

The construction of the Pick and Place manipulator is complex, mainly because of the required precision of operation. During the design and construction process of the machine, lessons were learned:

- When using the Mega2560 controller together with the RAMPS 1.6 overlay, it is not possible to directly control the LEDs. An additional Arduino Nano controller was required, which increased the cost of the design.
- The configuration of image processing algorithms requires frequent changes of lighting in order to select the most appropriate stages of its operation.
- The use of an off-the-shelf frame reduced the machine's construction costs, but it has a screw drive and is relatively heavy, resulting in low axle velocity, which increases the automatic stacking time.
- Mounting the manipulator to a table with wheels makes it easier to change the position of the machine, however, this makes it easier to cause shifting of elements placed in the working field, e.g. by accidental hitting the construction.
- In this article only the configuration for one SMD component was presented i.e. Capacitor 0805. For the full automatic placement of whole PCB project it would be necessary to make settings for each component.

The design of the manipulator allows for automatic placement of parts, but to achieve a reduction in machine time, several additional procedures can be

performed. The first step is to add and configure a second head. The mechanical design and electronic components of the robotic system have been adapted for this, which makes this process much easier. Another addition can be to construct automatic SMD component feeders. When these are mounted to the manipulator, it is possible to feed more components simultaneously, reducing the frequency of changing empty tapes. These would be controlled by using appropriate scripts.

The efficiency of the manipulator depends on the arrangement of PCBs on the working space and the arrangement of the feeders of SMD elements. The longer distance the head has to cover, the lower the productivity, however, after estimation the manipulator is able to place about 180–200 parts per hour.

References

[1] Andurkar G.K., Borkar V.: *Development of Pick and Place Robot for Industrial Applications*. International Research Journal of Engineering and Technology (IRJET), 4, 9, 2017.
[2] *Sneak Preview: DIY SMT Pick & Place Machine with OpenPnP*. https://mcuoneclipse.com/2018/05/05/sneak-preview-diy-smt-pick-place-machine-with-openpnp/ [15.12.2020].
[3] *Pick-and-Place Workflow using Stateflow for Matlab*. https://www.mathworks.com/help/robotics/examples/pick-and-place-workflow-using-stateflow.html [15.12.2020].
[4] *LitePlacer standard kit*. https://www.liteplacer.com/shop20/index.php?route=product/product&product_id=64 [18.12.2020].
[5] *The History of Industrial Robots, From Single Taskmaster to Self-Teacher*. https://redshift.autodesk.com/history-of-industrial-robots/ [15.12.2020].
[6] *JUKI RS-1R SMT Pick and Place Machine*. https://www. smt11.com/product/Pick-and-Place-Machine/JUKI-RS-1R-SMT-Pick-and-Place-Machine-78914.html#PRODUCT _CHARACTERISTICS [21.12.2020].
[7] *BARR group: Introduction to Pulse Width Modulation* (PWM). https://barrgroup.com/Embedded-Systems/How-To/PWM-Pulse-Width-Modulation [23.12.2020].
[8] *Coordinate System*. https://github.com/openpnp/openpnp/wiki/User-Manual#coordinate-system [29.12.2020].
[9] Evans M., Noble J., Hochenbaum J.: *Arduino w akcji*. Helion, Gliwice 2014: 124–125.
[10] *Setup and Calibration*. https://github.com/openpnp/openpnp/wiki/Setup-and-Calibration [3.01.2021].
[11] *Nozzle Tip Calibration Setup*. https://githubcom/openpnp/openpnp/wiki/Nozzle-Tip-Calibration-Setup [6.01.2021].
[12] *Kondensator 0805*. http://www.sm-elektronik.pl/galerie/1/10uf-6-3v-x7r-smd-0805-kondens_5882.jpg [9.01.2021].
[13] *Scripting*. https://github.com/openpnp/openpnp/wiki/Scripting [10.01.2021].

KRYSTIAN SZOPA, Ph.D., Eng.
PAWEŁ KARELUS, Eng.
Faculty of Mechanical Engineering and Robotics
AGH University of Science and Technology
al. A. Mickiewicza 30, 30-059 Krakow, Poland
kszopa@agh.edu.pl
pkarelus@student.agh.edu.pl