Article

Application of Computer Vision and Python Scripts as Educational Mechatronic Platforms for Engineering Development in STEM Technologies

Vladimir Tudić [1]*, Damir Kralj [1], Josip Hoster [1] and Tomislav Tropčić [2]

[1] Department of Mechanical Engineering, Karlovac University of Applied Sciences, 47000 Karlovac, Croatia; damir.kralj@vuka.hr (D.K.); josip.hoster@vuka.hr (J.H.)
[2] Student; tomislav.tropcic@gmail.com
* Correspondence: vladimir.tudic@vuka.hr

Abstract: This paper presents the process of designing, fabrication, assembling, programming and optimizing a prototype of a nonlinear mechatronic Ball-Plate System (BPS) as a laboratory platform for STEM engineer education. Due to the nonlinearity and complexity of BPS, task presents challenging issues, such as: 1) difficulties in controlling the stabilization of a given position point known as steady state error, 2) position resolution known as specific distance error and 3) adverse environmental effects - light shadow error, also discussed in this paper. The laboratory BPS prototype for education was designed, manufactured and installed at the Karlovac University of Applied Sciences at the Department of Mechanical Engineering, Study of mechatronics. The low-cost two degrees BPS system uses a USB HD camera for computer vision as feedback and two DC servomotors as actuators. Due to controlling problems, an advanced block diagram of control system is proposed and discussed. An open-source control system based on Python scripts that allows the use of ready-made functions from the library allows changing the color of the ball and the parameters of the PID controller, thus indirectly simplifying control system and directly the mathematical calculation. The authors will continue their research on this BPS mechatronic platform and control algorithms.

Keywords: Ball-Plate System; STEM; USB HD camera; Python scripts; ready-made functions; PID controller

1. Introduction

Undergraduate STEM engineering students require the practical application of theoretical concepts learned in the classroom in order to master controlling methods and issues. Our goal is to help students learn control theories of systems in an engineering context, through the design and implementation of simple and inexpensive BPS. Students are able to apply tools for computer modeling, control system design and software-hardware implementation in real time while solving the problem of ball position controlling. The entire project development is presented and can be assumed as a guide for replicating results or as a basis for a new approach to the design of mechatronic educational platforms. In both cases, we have at our disposal a tool for implementing and evaluating experimentally controlled strategies that can be further improved in the future. University laboratories and experiments play a very important role in successful education in STEM engineering, especially if the topic is robotics and automatic control applications. The rapid development of BPS applications has recently been noted due to the challenges related to controlling and fast dynamic response that requires short and fast sampling and instantaneous correction of the selected controller. Because controlling fast unstable systems is very important in a variety of practical applications, an educational mechatronic BPS platform can be a successful tool if used in training in robotics and automation control applications and control methods.

Feedback position of a ball is collected by using a camera as shown in [1]. Article is describing a controller synthesis for two-dimensional electromechanical system of the ball and plate intended for a study of the system dynamic and laboratory experiments with different proceeding of the control based on the classical and modern control theory. The system includes the quadratic metal plate, which is movably fixed in the center. Its inclination can be changeable in two rectangular directions. For the inclination of the plate a servo-drive with a controller and two stepping motors have been used. The control problem of the described system is to hold the freely rolling ball in the specific position on the plate. For measuring the ball position, the intelligent video system composed from CCD camera, picture-framing interface and program equipment for real-time picture processing are used.

The BPS was understood also as the two-dimensional movement of the ball and beam system presented in [2]. Author S. Awtar and others presented dynamic characteristics of the BPS, the mathematical model with the corresponding simplified model, and the analysis of the applications of different types of PID controllers. Based on the result of the analysis on different controllers, a controller using a switching mechanism is proposed to control the position of the BPS [3]. Also, F. Zheng in [4] includes hardware design description, sensor and actuator selection, system modeling, parameter identification, controller design and experimental testing.

The authors in [5] proposed a resistive touch screen technique to determine the position of a ball. This could successfully eliminate the lighting effect which can cause an error for camera-dependent control system. For the multivariable and complicated control system of BPS, a touch screen and a rotary pneumatic cylinder are chosen instead of a camera and a step motor, in this work. Simulation results show that it has good dynamic and static characteristics with the proposed control method. Not only fuzzy technique has become a popular choice for the BPS, some works utilized genetic algorithm with neural network or sliding mode controller to solve this nonlinear problem as shown in [6]. The PID neural network (PIDNN) controller based on genetic algorithm (GA) for BPS is proposed in this paper. GA is applied in training weighting factor of multilayered forward neural network, thus the disadvantage of backpropagation (BP) algorithm that easily fall into partial extreme value can be overcome and at the same time, the advantage of PIDNN controller that has simple structure and good dynamic, and static performance that is provided with the simulation results show that the proposed controller has the adaptability, strong robustness and satisfactory control performance in the BPS.

Furthermore, in paper [7] authors Y. Pattanapong and C. Deelertpaiboon proposes a position control technique for BPS using fuzzy logic with adaptive integral control action. The aim is that adaptive integral gain can automatically adjust its value linearly and only active when the ball's position is within the specified distance error. This novel scheme could utilize the ability of integral gain to eliminate steady-state error and adopts fuzzy logic technique because of its simplicity of not having to find mathematical model for this nonlinear system [8]. Current position of the ball is determined by using a webcam positioned right above the plate. Fuzzy controllers as advanced solutions are also described in [9,10]. In papers [11,12] the authors suggested sliding mode techniques (adaptive back stepping control) with the strategy of the fuzzy monitoring. They experimentally founded that adaptive sliding back stepping control is more effectively than the use of conventional SMC because it needs a lot of time to get a favorable tracking accuracy. Furthermore, one paper presents the usage of the neuro-controllers [13] and feedback linearization controllers [14]. Other papers consider advanced Model Predictive Control (MPC) [15].

Some paper presents a virtual and remote laboratory of the ball and plate system [16]. The authors in [17] proposed a control algorithm based on cascade PID and they compared it with another control method. The article shows the results of the accuracy of ball stabilization and influence of applied filter on the signal waveform. The application used to detect the ball position measured by digital camera has been written using a cross platform. Net wrapper to the OpenCV image processing library - EmguCV. The aim of the

article [18] is to help students learn the theory of control systems in an engineering context, through the design and implementation of a simple and low-cost ball and plate plant. Students are able to apply mathematical and computational modelling tools, control systems design, and real-time software-hardware implementation while solving a position regulation problem.

Numerous MPC algorithms have been used during recent history for various industrial process control but also used for numerous other processes; example applications are: heating, ventilation and air conditioning systems [19], robotic manipulators [20], electromagnetic mills [21], servomotors [22], quadrotors [23], autonomous vehicles [24], modular multirotor improved design of unmanned aerial vehicles [25, 26].

Finally, a new method was presented in [27]. Paper reveals an original BPS laboratory process; a simplified process model based on state–space process description. In this work a fast state–space MPC algorithm is discussed. According to authors its main advantage is computational simplicity: the manipulated variables are found on-line using explicit formulas with parameters calculated off-line; no real-time optimization is necessary.

Simplicity in the process of modeling the BPS system as well as avoiding complicated mathematical methods and formulas for stability state analysis led to the design and development of the mechatronic BPS prototype described in this paper. Aiming at achieve low-cost, simply and easy implementation and good controlling precision, this paper proposes computer vision as feedback, Python Open CV script PID controller with adjustable PID parameters to balance the ball for different determine position set points [28].

The contribution of this paper is summarized in several thematic sections:

1. Modeling of BPS prototype based on computer design skills for the purpose of fabrication all robotics and auxiliary parts.
2. Production of all designed parts using 3D printing technology and prefabricated aluminum square tubes to avoid machining of metal parts.
3. Usage of Python OpenCV script with ready-made functions instead of complex mathematical models and settings for a nonlinear system. Through the program code, a control algorithm is proposed and applied in accordance with the simplification of parameter manipulation by introducing the ready-made Python script functions.
4. Introduction of an interactive pop-up window for process manipulation; changing the color and set point of the ball position and controlling the coefficients of PID controller.
5. Discussion related to the proposed improved block diagram scheme.

The article is organized as follows. Section 2 briefly describes the computer design steps and procedures for making laboratory BPS prototype. The section includes the step of design individual robotic parts: servo motor shaft holders, levers and plate joints with as few parts as possible. The Python script algorithm is described in detail in Section 3 with an emphasis on description the ready-made functions involved in generating feedback by converting an image from a USB camera into a set of ball position correction request data. Section 4 deals with visualization and control; the pop-up window software implementation in relation to the HSV standard color palette settings and the PID controller coefficients settings. Section 5 presents an advanced block diagram scheme and discusses the results of experiments in which the proposed scheme is compared with the classical one. Also, part of the paper describes the discussion of the three indicated adverse effects that are perceived as errors in the conversion process and possible ways to eliminate or reduce them. Finally, Section 6 concludes the issues in the article.

2. BPS computer design and fabrication

This part of the paper discusses the steps of BPS design and production phase. The concept of BPS that was considered, designed and selected for production is actually a replica of similar BPS solutions listed in the works [1, 3, 7, 11, 16-18] basically, but in details

similar to [11, 27] (Figure 1). For the purpose of computer design, the software solution SolidWorks was accepted, and as printing software Ultimaker Cura. SolidWorks is well known as a software solution for computer-aided design (CAD) and computer-aided engineering (CAE) that is wildly used in all cases of technical and engineering design. Ultimaker Cura is the most popular printing software in the world. But first, the "driving board" for two servomotors or "step motors" was chosen. The logical choice was an Arduino UNO microcontroller board with two matching step actuators.

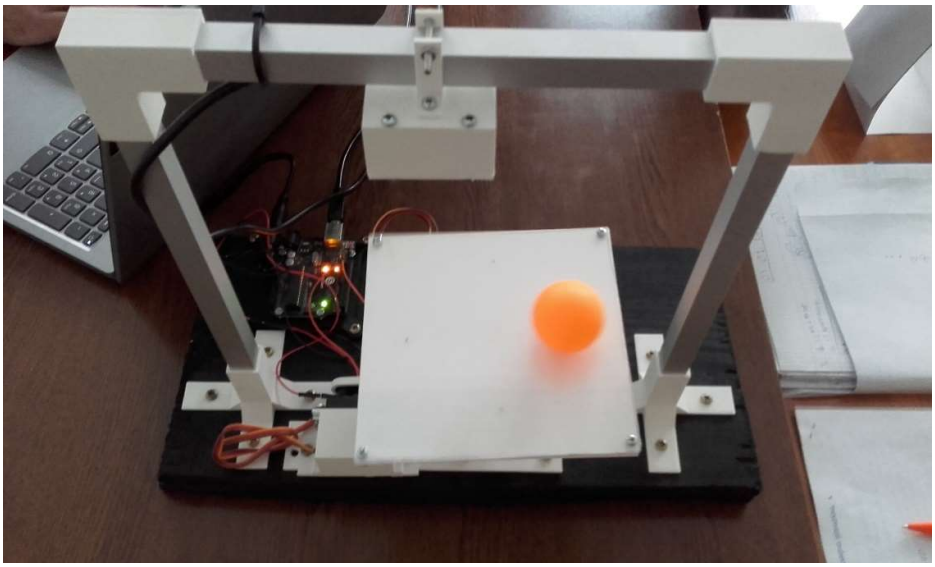


Figure 1. Laboratory BPS prototype in action.

2.1. Arduino UNO driving bord

Arduino UNO microcontroller board is used in this paper exclusively as a Pulse Width Modulation (PWM) driving platform for X-axis and Y-axis DC servomotors. Pulse Width Modulation is a technique which takes a constant steady state DC voltage and produces a train of fixed amplitude ON/OFF pulses whose average DC value is determined by the width or duration of the pulses given by the duty cycle. The current is determined by the impedance of the load being supplied. The Arduino's programming language makes PWM easy to use; by calling function or code analogWrite(pin, dutyCycle), where dutyCycle is a value from 0 to 255, and pin is one of the PWM pins (3, 5, 6, 9, 10, or 11). The analogWrite function provides a simple interface to the hardware PWM, but doesn't provide any control over frequency. It had to be noted that despite the function name, the output is a digital signal, often referred to as a square wave. As we have said before, at the Arduino Uno board, the PWM digital output pins are 3, 5, 6, 9, 10 and 11 as presented in [29]. The frequency of PWM signal on pins 5 and 6 are about 980Hz and on other pins are 490Hz. The PWM pins are labeled with "~" sign and for BPS controlling purpose pins 9 and 11 were selected. Stepper type actuators selected for this purpose was "Tower Pro MG995digi hi-speed" 3 wire, PWM controlled actuators. With operating speed of 0,2sec/60degree under 5 Volt power and metal gear transmission it is a very powerful actuator.

2.2. Computer design software

SolidWorks is a software solution for computer design and computer engineering wildly used in all technical and engineering design cases, as mentioned before, and incorporates the Finite Element analysis (FEM) Method. The FEM as computer code method provides a reliable numerical technique for analyzing engineering designs. The design process starts with the creation of a geometric model. Then, the program subdivides the

model into small pieces of simple shapes (elements) connected at common points (nodes). Finite element analysis programs look at the model as a network of discrete interconnected elements (mesh model). The Finite Element Method (FEM) predicts the behavior of the designed model by combining the information obtained from all elements making up the model.

Meshing is a crucial step in design analysis. The automatic mesher in the software generates a mesh based on a global element size, tolerance, and local mesh control specifications. Mesh control lets one specify different sizes of elements for components, faces, edges, and vertices.

The software estimates a global element size for the model taking into consideration its volume, surface area, and other geometric details. The size of the generated mesh (number of nodes and elements) depends on the geometry and dimensions of the model, element size, mesh tolerance, mesh control, and contact specifications. In the early stages of design analysis where approximate results may suffice, one can specify a larger element size for a faster solution. For a more accurate solution, a smaller element size may be required. Meshing generates 3D tetrahedral solid elements of one type, unless the mixed mesh type is specified [30].

Cura is 3D slicing software from Ultimaker that prepares a 3D model for printing, suitable for use with students at Upper KS2 and beyond. It is simple but powerful 3D slicing software. The print profiles are optimized for Ultimaker 3D printers, but the software will slice 3D files for any 3D printer brand/model. The software supports STL, 3MF and OBJ 3D file formats and also has a function that will import and convert 2D images (.JPG .PNG .BMP and .GIF) to 3D extruded models. The software will allow one to open and place multiple models on the print bed (each with different slicing settings if required). This allows users to print multiple models at a time, making classroom management of the printing process simpler [31].

2.3. Fabrication and mounting

Education BPS prototype because of simplicity consists of dozen parts; such as servomotor arm (Figures 2, 3), servomotor knee-arm (Figure 4, 5), servomotor housing (Figures 6, 7), BPS plate (Figures 8, 9), camera housing (Figure 10a), tube slippers (Figure 10b), base plate (Figure 11a), central pillar of BPS plate (Figure 11b), tube knees (Figure 12), Arduino board base plate and mounting screws. First the servomotor arm (Figures 2, 3) is designed with central elliptical hole for servomotor axle holder and smaller round hole for the arm bearing shaft. This connection must accept complete servomotor axle holder without any air clearness as well as the arm bearing shaft. The second part of servomotor, the knee-arm (Figures 4, 5) is designed and parameterized according to the actual size of the BPS plate for the same horizontal distances from the center of the plate ensuring identical angular transmission from the servomotor. Servomotor housing (Figure 6, 7) holds DC motor with screws to base plate (Figure 11a) in position. Central pillar of BPS plate (Figure 11b) with integrated small magnets makes a firm but flexible connection and assures central position of BPS plate (Figure 11b). Furthermore, both servomotor knee-arms also have integrated small magnets and in vertical position supports BPS plate in horizontal position (Figure 11b) firmly embracing the magnetic cups from the bottom of the BPS plate. Details of its construction are also given in student's undergraduate thesis [33].

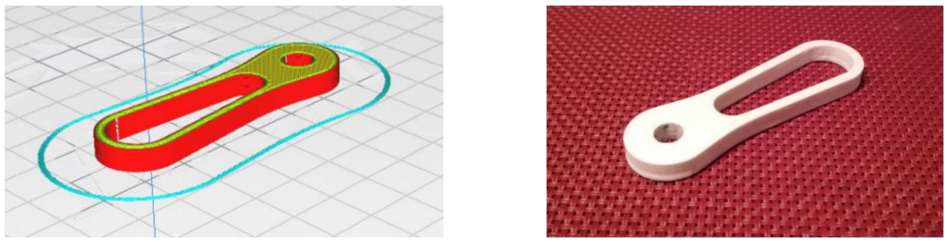
In the next few images (Figure 2 – 9), the design phases of some BPS parts are shown in SolidWorks software as the final files in Ultimaker Cura printing software.

Figure 2 shows the first robotic servo-arm of DC servomotor whose purpose is a strong connection to the original output of the DC servomotor shaft at one side (Figure 2a) and the spaceless joint of the shaft with the jaws of a knee joint on the second arm on the other side (small hole shown in the Figure 2a).

(**a**) (**b**)

Figure 2. Servomotor first servo-arm: (**a**) Visualization in SolidWorks; (**b**) SolidWorks mesh model.

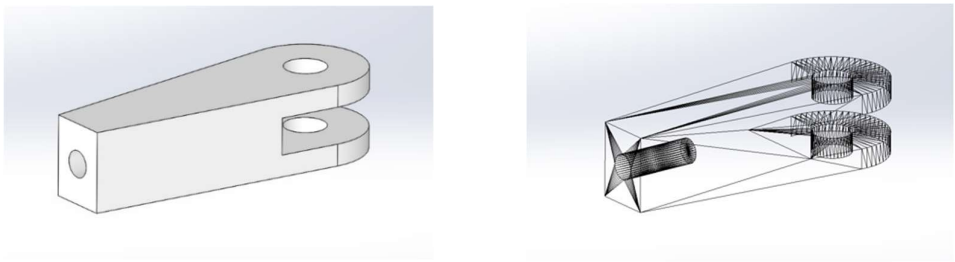
Figure 3 represents the first robotic servo-arm "slice phase" in the printing software and the finished part of the servo-arm after printing process.



(**a**) (**b**)

Figure 3. Servomotor first servo-arm: (**a**) Visualization in Ultimaker Cura software; (**b**) Actual 3D print.

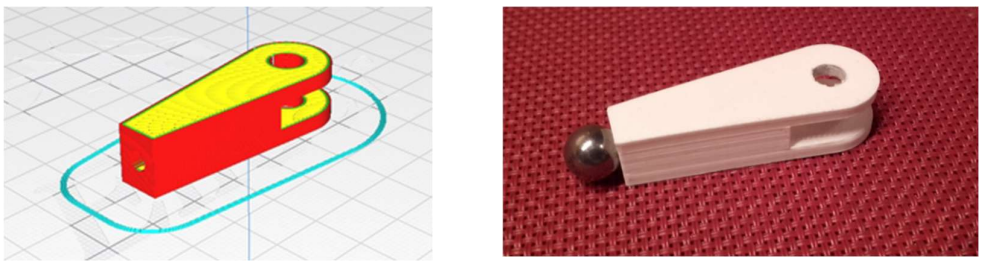
The crankshaft with the jaws of the second robotic arm of the servomotor is connected to the first servo-handle by inserting the shaft into a small hole (Figure 4) through both parts. The hole at the left side is a holder for a small magnet.



(**a**) (**b**)

Figure 4. Servomotor crank knee-arm: (**a**) Visualization in SolidWorks; (**b**) SolidWorks mesh model.

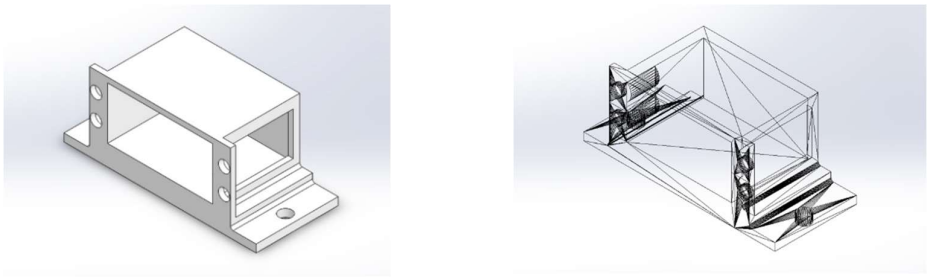
Crankshaft "slice phase" in the printing software and the finished part with installed magnet after printing process (Figure 5).



(**a**) (**b**)

Figure 5. Servomotor arm crank part: (**a**) Visualization in Ultimaker Cura software; (**b**) Actual 3D print.

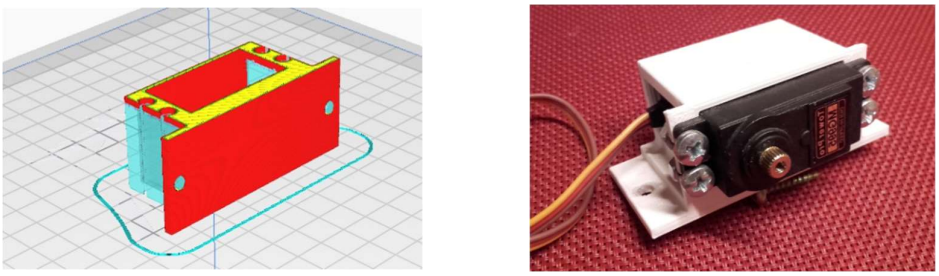
Tower Pro MG995 DC servomotor housing design phases (Figure 6).



(**a**) (**b**)

Figure 6. Servomotor housing: (**a**) Visualization in SolidWorks; (**b**) SolidWorks mesh model.

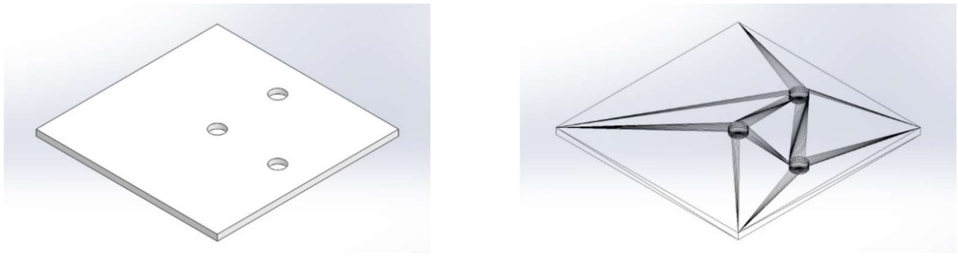
Tower Pro MG995 DC servomotor housing "slice phase" and finished part with built-in servomotor (Figure 7).



(**a**) (**b**)

Figure 7. Servomotor housing: (**a**) Visualization in Ultimaker Cura software; (**b**) Actual 3D print.

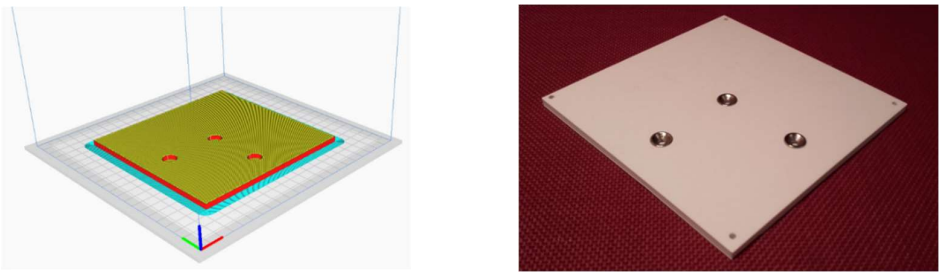
BPS plate housing design phases (Figure 8).



(**a**) (**b**)

Figure 8. BPS plate bottom view: (**a**) Visualization in SolidWorks; (**b**) SolidWorks mesh model.

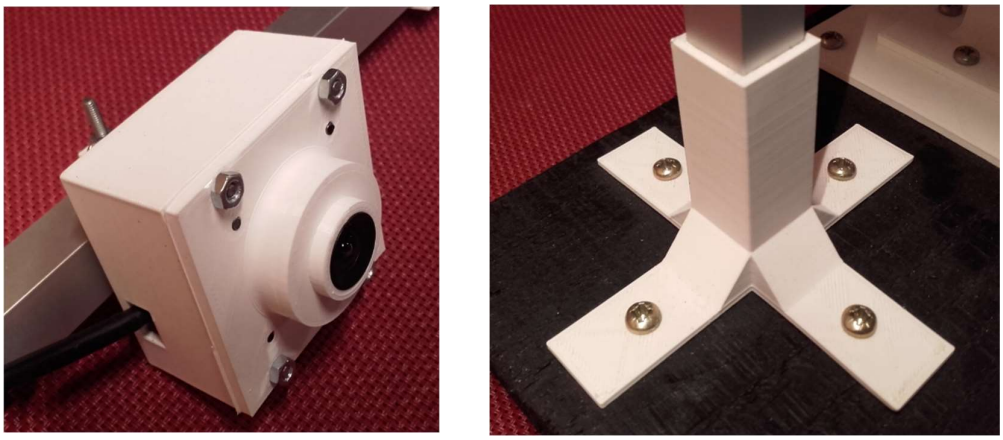
BPS plate "slice phase" and the finished part with installed magnetic cups (Figure 9).



(**a**) (**b**)

Figure 9. Bottom view of the BPS plate: (**a**) Visualization in Ultimaker Cura software; (**b**) Actual plate print with built-in magnetic cups for spaceless joint to the magnets.

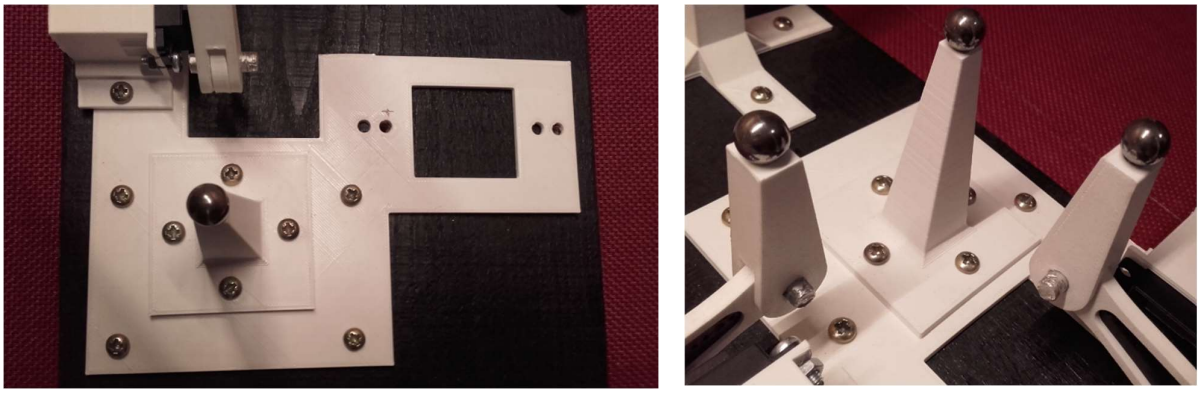
A USB camera built into the white housing is connected to a square tube (Figure 10a) and tube slippers for two vertical square tube pillars (Figure 10b).



(**a**) (**b**)

Figure 10. Printed BPS system parts: (**a**) USB camera housing; (**b**) USB camera tube slippers.

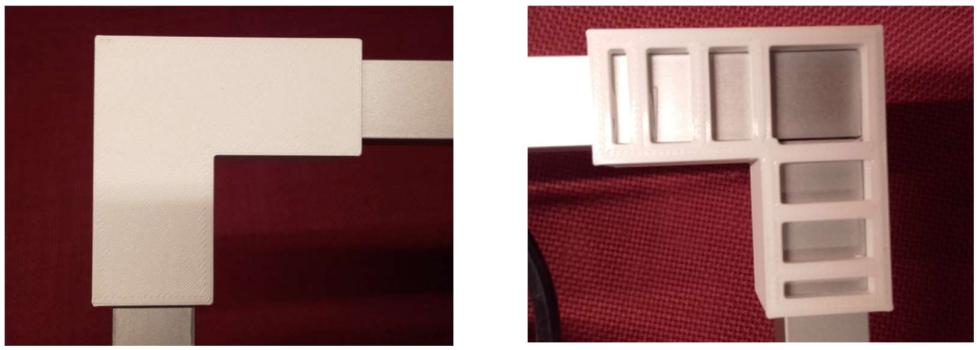
Base plate assembly for servomotors and central BPS pillar (Figure 11a) and three metallic balls for three magnetic cups under BPS plate (Figure 11b).



(**a**) (**b**)

Figure 11. Printed BPS system parts: (**a**) Base plate; (**b**) Three pillars; central fixed pillar and two vertical servomotor knee-arms.

Figure 12 shows the elbows for the horizontal and vertical mounting tubes for the camera holder.



(**a**) (**b**)

Figure 12. Printed square elbow for the camera holder tubes: (**a**) Front view; (**b**) Rear view.

2.4. General BPS design

In order for this paper to have a useful and practical application in student education, and not just a presentation of general theory, this part of the paper describes the problem of computer vision and its application in a mechatronic education. During the realization of the project, i.e. the preparation of the student's practical diploma thesis and later experimentation by the co-authors in this paper, some limitations and imperfections of the prototype and method were noticed, as well as problems in achieving stability of placing the ball in the given position. This paper and project are conceived as a simple and accessible experimental setup for learning, programming and understanding, the issues of feedback management in a real environment.

Mechatronic system described in paper was designed, developed and programmed with help of the student Tomislav Tropčić. The sideway of experimental platform is shown in Figure 13 (top left and right). The system uses a USB camera marked ELP-USBFHD01M fixed 200mm above the controlled platform on the camera holder (Figure 13). A designated 1920x1080 pixel (Full HD) camera that captures 30 frames per second. The technical designation of the camera is: ELP High Speed 120fps PCB USB2.0 Webcam Board 2 Mega Pixels 1080P OV2710 CMOS Camera Module with 2.1mm Lens ELP-USBFHD01M-L21.

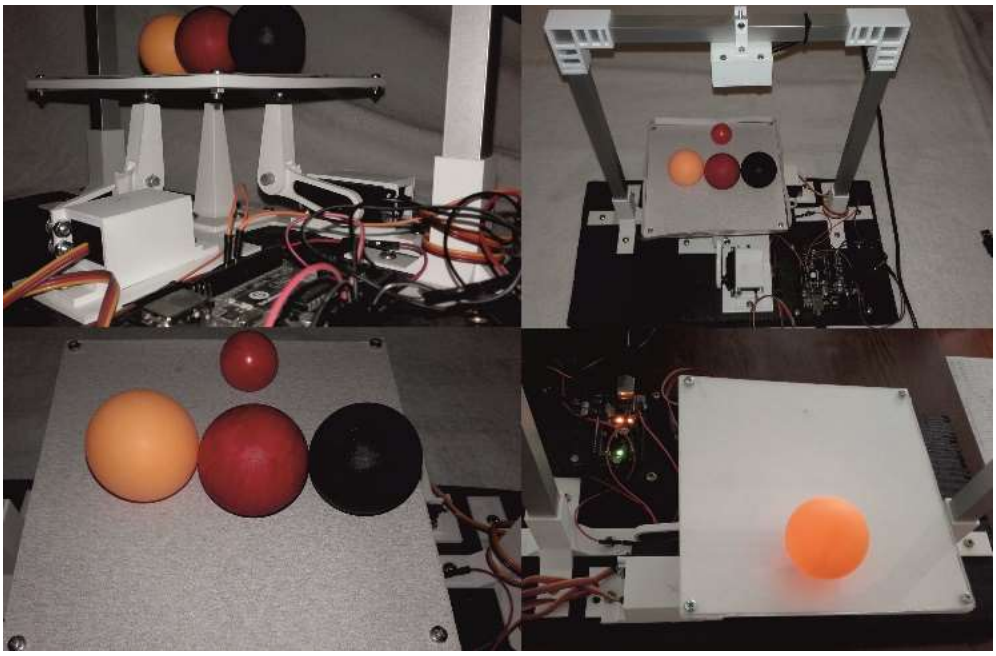


Figure 13. Education BPS platform with three experimental balls.

To perform the experiment, three balls of identical diameters but different colors were selected (Figure 13 bottom and left segment). Black, red and orange 40 mm diameter ball for table tennis were selected in turn. For comparison, a smaller red ball made of silicone mixture of significantly higher mass with a diameter of 20 mm was used. Several materials with the different friction property were used as the basis for the movement of the ball: 3D print material, Plexiglas, white paper and light grey sandpaper (sandpaper 180 particles per inch). The selected materials had different values of roughness and overtime gave unequal resistance during the movement of the test balls. The 150x150 mm white platform is placed on three supports, the middle of which is vertically motionless, and is placed in the geometric center of the square surface of the platform. In order for the platform to be tilted in both horizontal axes, a simple “magnetic” bearing with a ball and a cup on the underside of the platform in the geometric center has been designed. When

raising and lowering either of the two vertical robotic arms of the servomotor, the platform actually tilts in firm contact with the robotic handle on one side or the other. Servomotors are mounted with steerable arms with a wedge in the elbow (Figure 13 above and left) connected to the lower part of the steerable base. They are geometrically at an angle of 90 degrees to each other and the grips are equidistant from the central fixed bearing. The horizontal part of the servomotor handle (first arm) is attached to the servomotor protrusion and the vertical part of the handle (second arm with jaws) has a round metallic ball glued to the top in conjunction with a magnetic cup. Together they form a solid bearing that allows rotation because the cup is fixed in the lower part of the steerable base. With a solid elbow and a shaft with a wedge diameter of 4mm as a dry bearing, a simple robotic lever system has been created by which both servomotors with a rotary angle of ±15 degrees transmit the same angular displacement to the BPS platform.

3. Python script

3.1. Computer vision issues

One of the most important issues in the application of computer vision is performance in applications of recognizing patterns, shapes, colors and positions of objects. Given the limited amount of data typical for robotics, the challenges associated with selecting the appropriate substrate, lighting, and methods for assessing image and video quality without reference are important. An important aspect is that the algorithms used concern practical applications, not the derivation of mere theory, although simulations and visualization are important components in the preparatory phase of the scientific setup of an experiment.

Computational vision in mechatronics covers the following unavoidable topics: image formation, camera resolution, advanced image features, real-time sampling frequency, binary vision, optical flow, image filters, object creation, epipolar geometry reconstruction, motion tracking, segmentation, grouping, and recognition object. It is the possibilities of software modeling of image processing techniques and techniques for object localization and geometric measurements that enable advanced research in this scientific field. If the experimental setup uses a standard e.g., ELP-USBFHD01M camera software is becoming a powerful tool for evaluating and implementing image processing functionality.

3.2. Image converting technicques

The description of the ready-made functions used in the Python script related to image converting techniques is given in the order in which the image obtained using the USB camera is processed. In order to get more images per second, in the program code the resolution is halved to 640 x 480 pixels, so the number of captured images can be doubled, from 30 to 60 images per second. Ready-made Python image resolution function is defined as: “self.cam_width = 640, self.cam_height = 480”. The camera uses a USB connector to power and communicate with the computer.

3.3. List of Python ready-made functions

- VideoCapture object - VideoCapture()

When launching the application, it is necessary to create an object that will capture a video recorded with a USB camera. The application does not process the stored video (e.g. on the hard disk or memory card) but the stream of data that the camera records in real time (live stream), to download a series of images from the camera (30 images per second), the so-called VideoCapture object. The VideoCapture object only needs to specify the camera number (0=built-in, 1=external USB camera) where the recording comes from. Listing 1 shows a fragment of the code. All other processing (reception, processing and image

formation) will be performed autonomously "under the hood" of the ready-made function and thus free the programmer from a big job.

Listing 1: Fragment of the Python code: function cv2.VideoCapture(1).

```
def start_cam(self):
 self.capture = cv2.VideoCapture(1)
# 0 for webcam, 1 for external
```

In this part of the code, it is necessary to define the dimensions of the images captures by the camera, and it is defined that the image is 480 pixels high and 640 pixels wide.

- Color model conversion from RGB to HSV - cvtColor()

All colors are obtained by using and combining colors in the color palette. If we use the RGB (R-Red, G-Green, B-Blue) palette then we have 3 basic colors: red, green and blue. If each color is written in 256 shades, then by a combination of available shades we get a palette of 16.7 million colors. Another color representation (or color space) is HSV (H-Hue or Tone, S-Saturation, V-Value or Brightness). The paper uses the HSV palette, so the resulting RGB needs to be converted into HSV space (or model) colors. The switching was carried out in secret because it is easier to get a binary image of the object when it is written in HSV format be divided by subheadings. It should provide a concise and precise description of the experimental results, their interpretation, as well as the experimental conclusions that can be drawn (Listing 2).

- Noise image removal - GaussianBlur()

The next step is the process of removing noise from each image. The first step is blurring the edges of the image (Blur), using the Gaussian Blur function (blurring is performed using the Gaussian formula). When applied in two dimensions, this formula produces a surface whose contours are concentric circles with a Gaussian distribution from the center point. A matrix of certain dimensions is used to determine the degree of blur - the paper defines that the matrix has dimensions of 11x11 pixels (Listing 2).

- Binary image formation - inRange()

The captured image has a certain resolution (640x480 pixels), is converted to HSV color model and noise is removed. It is necessary to translate the image from a colored to a black and white image without shades - where the pixel in the image is colored with either black or white. It is necessary to determine which colors (and there can be 16.7 million in the picture) are converted to black and which to white (Listing 2).

- Binary image noise reduction - erode()

The resulting binary image may have certain noises that are usually located at the boundary of the contour of the object (in the binary image). Applying the erode() function of the application will remove certain noise, but the consequence may be a reduction in the contour of the object (Listing 2).

- Contour thickening - dilate()

In order to amplify the contour of the object on the binary image, it is necessary to use the dilate() function, which will "thicken" the contour by a certain amount of pixels in order to be more clearly visible (Listing 2).

Listing 2: Fragment of the Python code: other conversion function with associated parameters.

```
def process_frame(self, frame):
 self.hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
# transform into HSV color space
 blurred = cv2.GaussianBlur(self.hsv, (11,11), 0)
```

```
        mask = cv2.inRange(blurred, self.color_low, self.color_high)   # threshold
    mask = cv2.erode(mask, None, iterations=2)
    mask = cv2.dilate(mask, None, iterations=2)
```

- Object localization on a binary image – findContours()

After forming the binary image and the object, it is necessary to determine the contours of the object located in the image. The contours are passed to the application as a list of coordinates of the outer points that close the contour. There may be multiple contours in the image (intentionally, by mistake, or so) and then the application will look for the contour that occupies the largest area (Listing 3).

Listing 3: Fragment of the Python code: function findContours().

```
   # find countours in the mask
cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL,
# find the largest contour in the mask, then use
# it to compute the minimum enclosing circle
c = max(cnts, key=cv2.contourArea)
((x, y), radius) = cv2.minEnclosingCircle(c)
```

- Minimal circle within the contour - minEnclosingCircle()

After locating the contour of the object, the smallest circle is entered inside it so that the coordinates of the center and the size of the radius of the object can be determined. In this way, the center and edge of the contour on the binary image are determined (Listing 3). The procedure requires that the radius of the contour be a minimum of 10 pixels in length, and after finding the contour, the application displays a circle and its center so that the application user has an idea of where the application has located the center of mass or geometric center of the sphere (Listing 4). After determination, it is necessary to send the coordinates of the center of the contour and the radius according to the function that controls the PID controller self.PID(self.setPointX).

Listing 4: Fragment of the Python code: drawing setPoint on screen cv2.circle() and self.PID().

```
# only proceed if the radius meets a minimum size
if radius > 10:    # length od min 10 pixel
# draw setpoint on screen - 5 pixel red dot
cv2.circle(frame, (int(self.setPointX), int(self.setPointY)), 5,
(0, 0, 255), -1)
self.PID(self.setPointX, self.setPointY, x, y)
# PID setpoint actual position in x, y,
```

All used ready-made functions: VideoCapture(), cvtColor(), GaussianBlur(), inRange(), erodes(), dilates(), findContours(), minEnclosingCircle) and self.PID() in parentheses can receive certain parameter values. Each function does a lot of work (calculation) and significantly simplifies the application and its use. For this reason, the number of lines in the program and consequently the size of the control program is significantly reduced.

After running the script all functions and parameters are prepared to locate and calculate the ball shape and find its geometrical center as start set point (inputX, inputY). The Python script starts the motors and aligns the axes of the platform at the appropriate angles to align the stability with the initial start-position of the ball.

3.4. Python control script design

Python's control algorithm requires knowledge of past values. Proportional-integral control, for example, monitors the cumulative sum of differences between a setPoint and a process variable. Because the Python function disappears completely after feedback, the value of the cumulative sum must be stored elsewhere in the code. The problem with coding is figuring out how and where to store this information between call algorithms. For coding reasons, an object generator was created where certain parameter values can be received in parentheses. There are several ways to get value from a so-called number generator. One way is to use the next() function which executes the generator until the next yield expression is encountered and then returns the value.

Python script captures series of camera images at 30 frames per second, approximately every 33 milliseconds, which is the sampling rate of the ball position or the speed of calculating the position correction. Thus, the parameter dT is a time constant that correlates with the image processing speed, which is a PID controller feedback parameter. Listing 5 shows the code fragment where the time variable was defined.

Listing 5: Fragment of the Python code: time variable dT definition.

```
# how long since we last calculated (dT definition)
now = time.time() # now = begining of application
# change in time (dT = )
  dT = now - self.last_time     #print(dT)
  # save for next iteration
  self.last_time = now
```

3.5. Advanced block diagram of PID controller

During the experiment and the selection of the most suitable color, shape and size of the ball as well as the surface of the plate, it was realized that the block diagram is not as simple as it seemed at first glance. Significant and unavoidable disturbances were observed, i.e., external influences that prevented the stable operation of the mechatronic system and the placement of the ball at a given setPoint. Observed interference functions that cannot be accurately described mathematically, but have been shown to be influential because methods of reducing problems and attempt to eliminate them have led to better results and greater stability. For this reason, an improved block diagram control loop was proposed (Figure 14) that highlights the locations in the CLC loop and the type of dysfunction or detrimental effect on ball position stabilization. First of all, a dysfunction (accidental disorder) is defined, which is denoted as $d_1(t)$, which represents mechanical imperfections and clearances of the handles that contribute to the increase of error.

Furthermore, dysfunction $d_2(t)$ describes a group of functions within software that, if inconsistent or unable to perform their task properly, increase position vagueness and introduce uncertainty and directly lead to significant problems and instabilities during position control.

The third influential quantity that contributes the most to the results of the experiment is the amount of scattering or light intensity. The system has been shown to have the greatest stability if light is scattered on the substrate from several sources and the original beam of the lamp is shaded. Each shadow from the light source significantly changes the

color shade of the ball on the edge of the ball and changes the contour image, which contributes to poorer recognition of the contour shape and consequently the creation of a binary image. It has been observed that with a single light source although the system has dispersive structure, controller cannot stabilize the ball at all due to the above errors or conversion imperfections. In block diagram view, dysfunction $d_1(t)$ has a direct impact on the process (plate position) and form "steady state error". Similarly, dysfunction $d_2(t)$ as "internal" uncertainty creates a cumulative effect on the Python output data set (inputX, inputY) before the setPoint calculation process (setPointX, setPointY) and thus forms a "light shadow error".

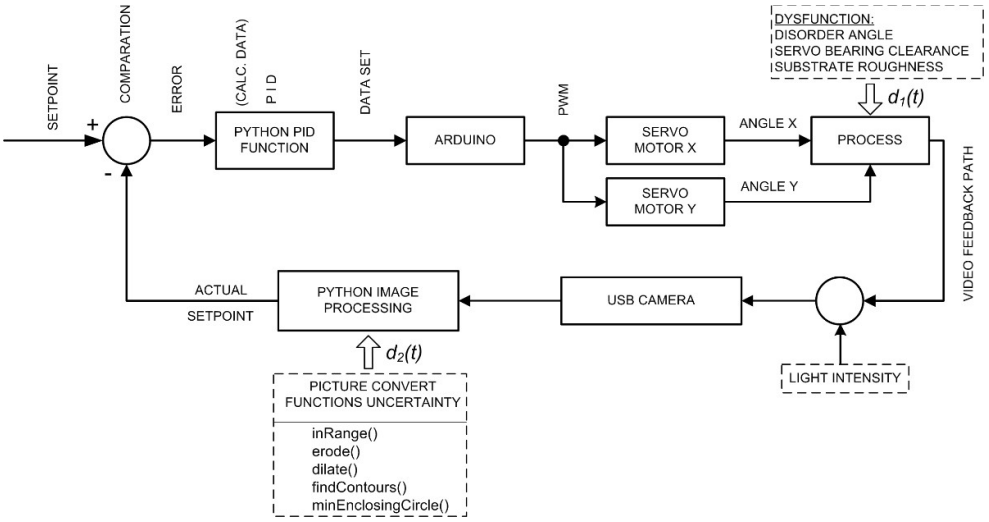


Figure 14. Advanced control diagram scheme.

3.5. CLC Error value calculation

The equations built into the Python script make strategic progress, emphasize the ability to create program calculations and manage without real physical hardware (external controller). Error functions for both x and y axes (errorX, errorY) come from the comparison process and are defined in the program code by parameters (Listing 6).

Listing 6: Fragment of the Python code: calculation of the error values.

```
# error variables
errorX = self.setPointX - inputX
errorY = self.setPointY - inputY
#print(errorX, errorY)
```

The PID control system generally consists of two independent classic PID controllers with one loop. The first of them controls the x position of the ball by manipulating the PWM signal of the first DC servomotor. The second controls the y position using the PWM signal of the second DC servomotor. The PID controller used the same coefficients for both axes assuming the board has two axes uniformity and ideal perpendicularity.

Listing 7: Fragment of the Python code: calculation of both PID outputs for angle correction.

```
# angle variables
angleX = self.zero_x + (errorX * self.kP + dErrorX * self.kD +
self.kI * self.errorSumX)
```

```
angleY = self.zero_y + (errorY * self.kP + dErrorY * self.kD +
self.kI * self.errorSumY)
```

The coefficients of the PID controller *kP*, *kD* and *kI* were selected in the optimization process and entered into the program as default values, however, they can be changed during the experiment in the control application pop-up window.

Listing 8: Fragment of the Python code: PID controller coefficients default values.

```
# PID coefficients
self.default_P = 0.033
self.default_D = 0.023
self.default_I = 0.001
```

Proportional *kP* is used to find out the error between the desired value and actual value and is responsible for the corrective response (Listing 7). Integral *kI* is applied to calculate all the past values of error and then integrate them to find out the Integral term. When error is expelled from the system this integral stops increasing. The derivative *kD* is used to predict the expected error values in future based on the present values. Controlling effect can be increased if the system has a rapid rate of change, which is also based on Derivative part. Combining all these three operations gives the total value of the required correction. The constants *kP*, *kI* and *kD* of the PID controller can be entered by the program code but also changed in the graphical visualization window presented in Figure 15.

Calculated output data set for Arduino controller as PWM driving platform for both servomotors commands, is given below (Listing 9).

Listing 9: Fragment of the Python code: formatted values for the Arduino board.

```

# send to arduino
arduino.write((str(angleX) + "," + str(angleY) + "\n").encode())
# print(angleX, angleY)
```

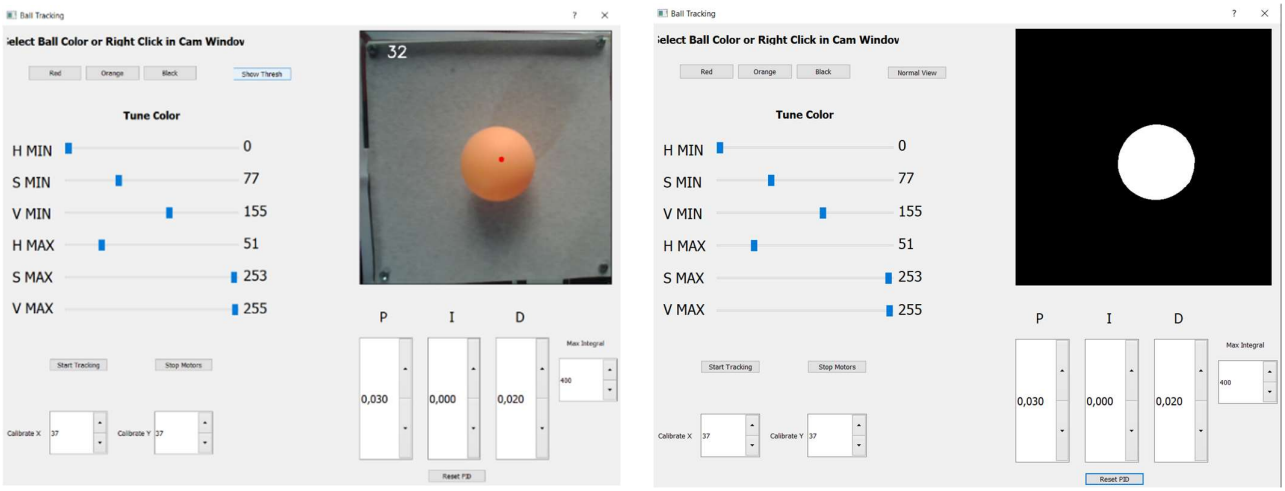
4. Visualization and control

In this part of the paper, a discussion focused on visualizing the position of the ball after activating the application and managing the position of the ball. First, during the experiment, it was proved that of the three selected balls, the highest quality conversion to a binary image and the entire image processing covers the case of modified orange color (HSV format parameters - 0/77/115/51/253/255) with a slight deviation from the entered "default" value in relation to the value entered in program code (HSV - default 1/77/115/61/153/255). The red color (HSV - default parameters 121/157/86/243/255/255) did not give sufficient response quality, despite of parameter modification, which resulted in an increase in the value of the disturbance function $d_2(t)$ and ultimately too much error and deviation in the calculation, which manifested itself as the possibility of setting the red ball to a given default set point on the platform. The black ball, despite having the strongest color contrast in its parameters (HSV-default 0/0/0/25/25/25), could not be recognized at all as a shape or contour in the HSV standard, probably due to poor lighting quality.

4.1. Application "Ball Tracking"

The interactive "Ball Tracking" pop-up window shown in Figure 15 is designed as the controlling device window of the mechatronic BPS prototype. With the help of designed functions that are performed on the screen, it is possible to control the process with various important parameters. The small white dot, 5 pixels wide, represents the center of mass calculated in Python script as the actual center of the orange ball. Ball tracking window is designed to contain two useful presentations of ball images in the upper right corner, the normal and the calculated binary variant (Figure 15a and Figure 15b). Although the small white dot represents the center of the ball, clicking on a new position on the board on the computer screen, determines the desired position of the ball as a small red dot, also 5 pixels wide. The equations for calculating the PID error values in a Python script automatically calculates the correction value for both axes and thus balance the BPS plate with both actuators ([https://youtu.be/8cQ3oVXJYzA]).

On the left side of Ball Tracking pop-up window are six HSV pallet sliders that allow fine tuning of color shades for the ideal conversion of binary presentation ("Show Thresh"). Figure 15a, in the upper right corner, is a real-time image from a USB camera with an orange ball that showed the best response and presentation of the image live stream during the experiment. Below this section you can see three frames for fine-tuning the controller PID coefficients in steps of 0.001 units or the "Reset PID" option for the default values (stored in script). In the lower left corner are two square buttons for controlling object search (Start/Stop Tracking) and the right button for controlling servomotors (Start/Stop Motors). Furthermore, at the very bottom of the interactive window there is a very useful option to further adjust the horizontality of the plate with respect to the unevenness of the substrate on which the prototype is located. Also, the sampling frequency of the image in millisecond is shown in the upper left corner (insert value 32 in Figure 15a). In Figure 15b, a binary figure of ideal shape shows the same position of the ball. Changing images is allowed by pressing the "Show Thresh"/" Normal View" button.



(**a**) (**b**)

Figure 15. "Ball Tracking" pop-up window: (**a**) "Normal View" shows actual camera live stream image; (**b**) "Show Thresh" shows binary image (white ball contour on the black plate).

The mechanical "zero position" of the plate, as for the robotic arms of servomotor, is at a default angle of 37-degree for both actuators (Figure 15a and Figure 15b). The "zero position" can be set in 1-degree steps increments within the "Calibrate" X and Y axis frames. Angle control on both axes is limited to ±15 degrees.

In the Python script, two more pop-up screens have been created that indicate the graphical representation of the relevant parameters as well as their numerical matrix representation for future mathematical analyzes. One is 6 seconds time period diagram with a graphical presentation of the actual and selected position set point, and shaft angle value

as PID correction response. The second manageable pop-up window in the Python code represents numerical data of parameters in the same time period.

5. Experiment results

After making the prototype, it was necessary to functionally test the work and optimize all the functions of the Python script with actual components of the BPS prototype. After several iterations, the functional operation of the BPS system was obtained, which enabled the setting of a ball setPoint anywhere of the surface of the plate (150x150mm). But, a "sliding" of the smooth ball on a smooth Plexiglas plate substrate was observed. This is a proven imperfection of the mechanical system and the plate substrate, commonly referred to as "steady-state error" or dysfunction $d_1(t)$, as shown in the advanced block diagram (Figure 14). During experiment typical "overshooting" error values are in domain of ±24 pixels (approx. ±8 mm).

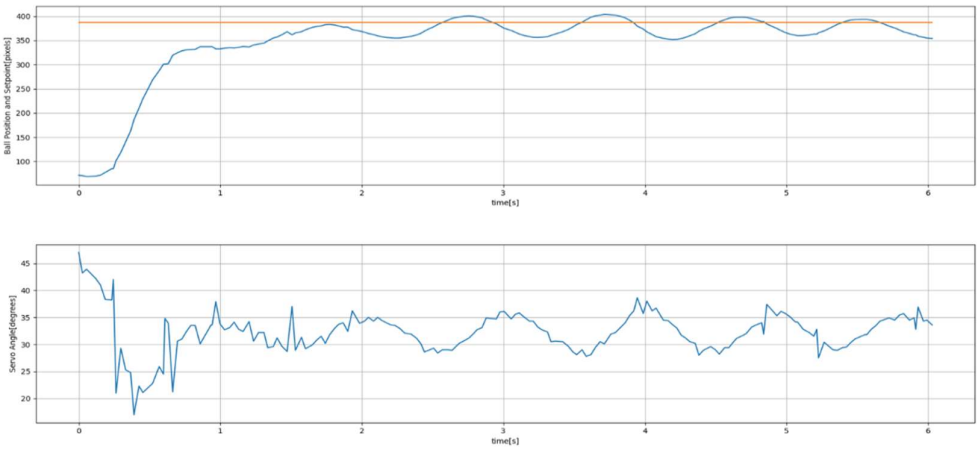


Figure 16. Top graph: actual ball position=40 pixel (blue) and setPoint position=390 pixel (red) of the ball after X-axe setPoint shift: 350 pixels (graph time 0s): "steady-state error" of ±24 pixels (±8 mm) caused by "light shadow error". Lower graph: X-axes servomotor angle corrections.

Furthermore, during the experiment, a large influence of lighting on the operation of the system was observed. Strong light sources on one side of the ball have been shown to prevent the system from bringing the ball to its desired location. This is a proven feedback imperfection due to an inaccurate image conversion system called "light shadow error" or $d_2(t)$ dysfunction in the paper, which is also shown in the advanced block diagram. To counteract this detrimental effect, several smaller discrete shaded light sources were included and then the control accuracy was significantly improved. Figure 17 shows "light shadow error" at the right side of orange ball contour diameter enters physical ball border and binary image is disrupted.

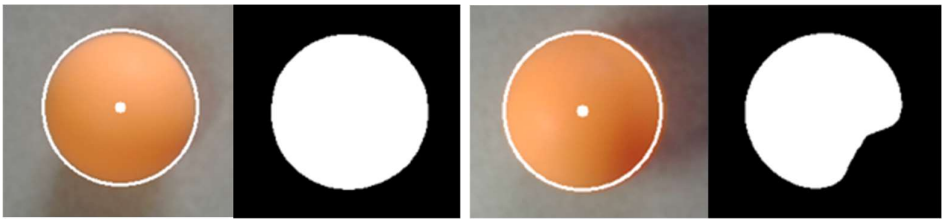


Figure 17. Orange ball tracking: left-ideal conversion, right-"light shadow error".

Furthermore, a resolution problem was observed, called "specific distance error": the inability of the system to recognize a new set point of the sphere center position near the existing actual set point. A "specific distance error" has been shown to be the length of a red dot radius of 6 pixels (2 mm) that marks the center of mass of a sphere (Figure 18).

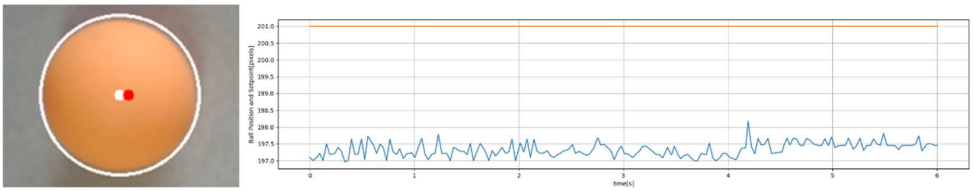


Figure 18. "Specific distance error", X-axis setPoint offset: 4 pixels, but no controller response.

Also, Figure 19 shows the "specific distance error" after controller correction process: the setPoint change 12 pixels (4 mm) in the left X-axis direction. Stabilization can be seen after 2 seconds of control and residual dislocation of the actual ball position approximately 2 pixels (0.7 mm) away. Lower graph at Figure 19 shows X-axe servomotor angle correction; after 2 seconds signal noise observed in range ±0,2 degree.

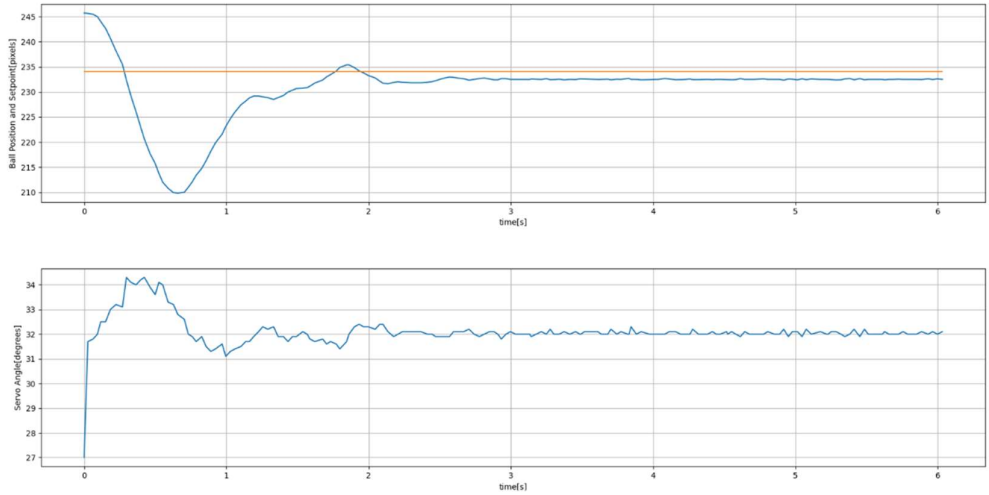


Figure 19. Top graph: actual ball position=246 pixel (blue) and setPoint position=234 pixel (red) at time 0s. Lower graph: X-axe servomotor angle correction.

To increase the roughness of surface of the plate, sandpaper with a granulation of 180 points was placed. With this background, controlling has been significantly improved. Also, one can see controller order to actuators for correction every 32 milliseconds (lower image on Figure 20).

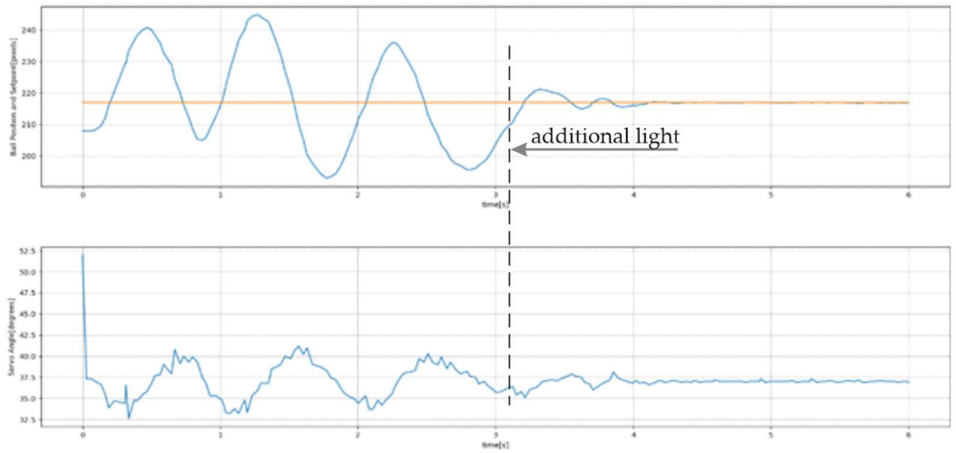


Figure 20. Introduction of an additional light source; ball stabilization after 1 seconds.

In order to demonstrate the effect of the amount of input light on the feedback efficiency, an additional light source on the other side of the panel was switched on in the third second of the experiment. Improved lighting led to the instantaneous stabilization of the ball as shown in Figure 20. Also, in the first three seconds there was a delay in response or evident sluggishness of the ball, which is not the focus of this paper and which will certainly be taken into account in future scientific research.

5. Conclusions

This paper discusses the application of the BPS prototype as a laboratory platform for the education of STEM engineers. Moreover, details on the design and implementation of software and hardware it is also discussed. An open-source control system based on Python scripts that allows the use of ready-made functions from the library is characterized by a very short computation time. Simplicity of the calculation is possible due to the OpenCV environment. When applied to the BPS process, the OpenCV algorithm was considered to work, but compared to other papers it is necessary to improve the system to eliminate or partially reduce the impact of size disturbances indicated as errors in the improved block diagram.

Therefore, the authors hope that the presented works may be inspiring for the readers, students, leading them to further development of new methods and applications of machine vision and computer vision methods for industrial and non-industrial purposes, because the authors will certainly continue their research on this BPS mechatronic platform and control algorithms.

Author Contributions: Conceptualization, V.T., J.H. and D.K.; methodology, V.T., J.H.; software, T.T.; validation, V.T., and D.K.; formal analysis, D.K., J.H.; investigation, T.T.; resources, T.T.; data curation, V.T.; writing—original draft preparation, V.T.; writing—review and editing, D.K.; visualization, V.T., T.T.; supervision, D.K, J.H.; project administration, V.T. All authors have read and agreed to the published version of the manuscript.

Funding: This paper was produced as part of the "Atrium of Knowledge" project co-financed by the European Union from the European Regional Development Fund and the Operational Programme "Competitiveness and Cohesion" (OPCC) 2014 - 2020. Contract No: KK.01.1.1.02.0005.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Knuplez, A.; Chowdhury, A.; Svecko, R., Modeling and Control design for the ball and plate system, in Proc. of IEEE Int. conf. on Industrial Technology, December 2003, pp. 1064–1067, DOI: 10.1109/ICIT.2003.1290810.
2. Hongwei, L.; Yanyang, L., Trajectory tracking sliding mode control of ball and plate system, in Proc. of 2nd Int. Asia Conf. on Informatics in Control, Automation and Robotics, March 2010, pp. 142–145, DOI**:** 10.1109/CAR.2010.5456649
3. Awtar, S. et al, Mechatronic design of a ball-on-plate balancing system, *Mechatronics*, vol. 12, no. 2, March 2002, pp. 217–228, https:// doi.org/10.1016/S0957-4158(01)00062-9
4. Zheng, F.; Li, X.; Wang, S.; Ding, D., Position Control of Ball and Plate System Based on Switching Mechanism, in Proc. of IEEE Int. conf. on Automation and Logistics, August 2011, pp. 233–237, DOI: 10.1109/ICAL.2011.6024719
5. Matsuo, T.; Tsuruta, K.; Suemitsu, H., Fuzzy adaptive identification method based on Riccati equation and its application to ball-pate control system, in Proc. of IEEE Int. conf. on Systems, Man, and Cybernetics, October 1999, pp. 162–167, DOI: 10.1016/j.isatra.2011.01.014
6. Dong, X.; Zhang, Z.; Chen, C., Applying genetic algorithm to on-line updated PID neural network controllers for ball and plate system, in Proc. of IEEE Int. conf. on Innovative Computing, Information and Control, December 2009, pp. 751–755, DOI: 10.1109/ICICIC.2009.113

7. Pattanapong, Y.; Deelertpaiboon, C., On Ball and plate position control based on fuzzy logic with adaptive integral control action, *Mechatronics and Automation* (ICMA), 2013 IEEE International Conference, August 2013, pp. 1513–1517, DOI: 10.1109/ICMA.2013.6618138
8. Kassem, A.; Haddad, H.; Albitar, C., Commparison Between Different Methods of Control of Ball and Plate System with 6DOF Stewart Platform, IFAC-PapersOnLine 2015, 48, pp. 47–52, https:// doi.org/10.1016/j.ifacol.2015.09.158
9. Morales, L.; Gordón, M.; Camacho, O.; Rosales, A.; Pozo, D., A Comparative Analysis among Different Controllers Applied to the Experimental Ball and Plate System, In Proceedings of the 2017 International Conference on Information Systems and Computer Science (INCISCOS), Quito, Ecuador, 23–25 November 2017; pp. 108–114, DOI: 10.1109/INCISCOS.2017.27
10. Robayo Betancourt, F.I.; Brand Alarcon, S.M.; Aristizabal Velasquez, L.F., Fuzzy and PID controllers applied to ball and plate system, In Proceedings of the 2019 IEEE 4th Colombian Conference on Automatic Control (CCAC), Medellin, Colombia, 15–18 October 2019; pp. 1–6, DOI: 10.1109/CCAC.2019.8921113
11. Bdoor, S.R.; Ismail, O.; Roman, M.R.; Y. Hendawi, Design and Implementation of a Vision-based Control for a Ball and Plate System, 2nd International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM), 2016, **DOI:** 10.1109/ICIEAM.2016.7910965
12. Jeon, J.H.; Hyun, C.H., Adaptive Sliding Mode Control of Ball and Plate Systems for Its Practical Application, 2nd International Conference on Control and Robotics Engineering, April 2017, DOI:10.1109/ICCRE.2017.7935054
13. Moreno-Armendáriz, M.A.; Pérez-Olvera, C.A.; Rodríguez, F.O., Indirect hierarchical FCMAC control for the ball and plate system, *Neurocomputing* 2010, 73, pp. 2454–2463, https:// doi.org/10.1016/j.neucom.2010.03.023
14. Huang, W.; Zhao, Y.; Ye, Y.; Xie, W., State Feedback Control for Stabilization of the Ball and Plate System, In Proceedings of the 2019 Chinese Control Conference (CCC), Guangzhou, China, 27–30 July 2019; pp. 687–690, 10.23919/ChiCC.2019.8866355
15. Tatjewski, P., Disturbance modeling and state estimation for offset-free predictive control with state-space models, Int. J. *Appl. Math. Comput. Sci.* 2014, 24, pp. 313–323, 10.2478/amcs-2014-0023
16. Fabregas, E., Dormido-Canto, S., Dormido, S., Virtual and Remote Laboratory with the Ball and Plate System, IFAC-PapersOnLine,Volume 50, Issue 1, July 2017, pp. 9132-9137, https:// doi.org/10.1016/j.ifacol.2017.08.1716
17. Linder, T.; Rybarczik, D.; Wirwal, D., Stabilisation problem in biaxial platform, June 2016, *Archives of Mechanical Technology and Materials* 36(1), DOI: 10.1515/amtm-2016-0012
18. Stander, D.; Jiménez-Leudo, S.; Quijano, N., Low-Cost "ball and Plate" design and implementation for learning control systems. In Proceedings of the 2017 IEEE 3rd Colombian Conference on Automatic Control (CCAC), Cartagena, Colombia, 18–20 October 2017; pp. 1–6, 10.1109/CCAC.2017.8276472
19. Carli, R.; Cavone, G.; Ben Othman, S.; Dotoli, M., IoT Based Architecture for Model Predictive Control of HVAC Systems in Smart Buildings. *Sensors*, 2020, 20, pp. 781, https:// doi.org/10.3390/s20030781
20. Rybus, T.; Seweryn, K.; S ˛asiadek, J.Z, Application of predictive control for manipulator mounted on a satellite, *Arch. Control Sci.* 2018, 28, pp. 105–118, DOI: 10.24425/119079
21. Ogonowski, S.; Bismor, D.; Ogonowski, Z., Control of complex dynamic nonlinear loading process for electromagnetic mill, *Arch. Control Sci.* 2020, 30, pp. 471–500, DOI 10.24425/acs.2020.134674
22. Horla, D., Experimental Results on Actuator/Sensor Failures in Adaptive GPC Position Control, *Actuators* 2021, 10, pp. 43, https:// doi.org/10.3390/act10030043
23. Eskandarpour, A.; Sharf, I., A constrained error-based MPC for path following of quadrotor with stability analysis, *Nonlinear Dyn.* 2020, 98, pp. 899–918, DOI: 10.1007/s11071-019-04859-0
24. Ducajú, S.; Salt Llobregat, J.J.; Cuenca, Á.; Tomizuka, M., Autonomous Ground Vehicle Lane-Keeping LPV Model-Based Control: Dual-Rate State Estimation and Comparison of Different Real-Time Control Strategies, *Sensors,* 2021, 21, pp. 1531, https:// doi.org/10.3390/s21041531
25. Kotarski, D.; Piljek, P.; Kasać, J.; Majetić, D., Performance Analysis of Fully Actuated Multirotor Unmanned Aerial Vehicle Configurations with Passively Tilted Rotors, *Applied Sciences-Basel,* **11** (2021), 18; pp. 8786, 18, https:// doi:10.3390/app11188786
26. Kotarski, D., Piljek, P.; Pranjić, M.; Giorgio Grlj, C.; Kasać, J., A Modular Multirotor Unmanned Aerial Vehicle Design Approach for Development of an Engineering Education Platform, *Sensors,* **21** (2021), 8; 2737, pp. 24, https:// doi:10.3390/s21082737
27. Zarzycki, K.; Ławrynczuk, M., Fast Real-Time Model Predictive Control for a Ball-on-Plate Process. *Sensors* 2021, 21, pp. 3959. https:// doi.org/10.3390/s21123959.
28. Available online: https://docs.python.org/3/ (accessed on 5 December 2021).
29. Available online: https://create.arduino.cc/projecthub/muhammad-aqib/arduino-pwm-tutorial-ae9d71, (accessed on 8 December 2021).
30. Available online: https://help.solidworks.com/2020/english/SolidWorks/cworks/c_Background_on_Meshing.htm (accessed on 9 December 2021).
31. Available online: https://www.createeducation.com/software/cura/ (accessed on 10 December 2021).
32. Available online: https://www.rccorner.ae/towerpro-mg995-digi-hi-speed-servo (accessed on 4 December 2021).
33. Tropčić, T., Application of Computer Vision in Mechatronic, Undergraduate Thesis, Karlovac University of Applied Sciences, 10 September 2020, https://urn.nsk.hr/urn:nbn:hr:128:892674 (accessed on 20 January 2021).
34. Available online: https://www.youtube.com/watch?v=LADO4qKQaGc (accessed on 15 December 2021).