The application of VPython to visualization and control of robot

Maciej Wochal, Dawid Cekus, Pawel Warys

Czestochowa University of Technology

Abstract: The idea of using VPython software environment (Python programming language plus a 3D graphics module called Visual) for visualization and control of robot movements is presented in the paper. The control system has been worked out for walking robot, which is building at Institute of Mechanics and Machine Design Foundation of Technical University of Czestochowa. A method of modeling and programming of robot operation control system has been described. The used library to communication with external devices allows easy control of an arbitrary number of mobile robotic modules. The developed program allows for cooperation with any servo controller and enables work in simulation mode of system motion or control mode. In control mode, the control sequences are sent in real time to the executive system and realized movements are shown in the form of three-dimensional visualization on a computer screen. User has the possibility of dynamic defining the position of the separate or group of elements in the space, and the change of velocity and position.

Keywords: walking robot, control system, Python

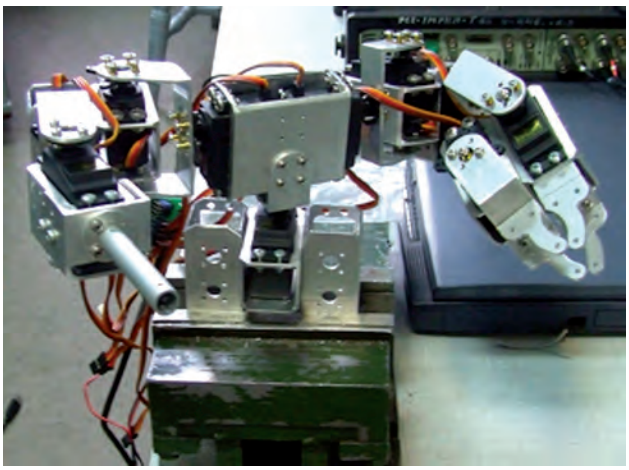

Fig. 1. Walking robot under construction

Rys. 1. Budowany robot kroczący

1. Introduction

This paper concerns the visualization of movements working (the actual position of robot components) and method for controlling of walking robot, which is built as a part of Scientific Society of Computer Aided Design of Mechatronic Devices and Machines, and its main aim is the presentation the Python programming language and libraries used for implementation of these tasks.

On the current stage of work, the robot can be used as manipulator (fig. 1), but in the future, the robot will be moved on two legs.

In Poland, as well as over the world a lot of scientific and research centers deal with the issues concerning the walking robot [1–4]. In the paper [4] the problem of conception and building 4-legged walking robot intended for transport cargo was presented. Starting from the structural synthesis of leg the kinematic chain was developed, and the computer calculation model was built. Considerations have been taken to develop programs and walking algorithms (straight, a curve, by obstacles). The analysis was finished by simulations testing the robot movement in order to verify the correctness of the solutions assumed.

Constructional solutions of walking robots are based mainly on solutions observed in nature in humans, animals, or insects. One example of this might be a solution where the machine uses a tail to keep one's balance. Analysis of the gait stability of a man moving along an even surface with a constant velocity is presented in article [5]. The stability criteria applied to biped robots, namely: the Zero Moment Point (ZMP) and the Ground projection of the Center Of Mass (GCOM) have been employed in the investigations. The analysis has been carried out on the basis of measurement data obtained from the human gait recorded with a digital camera.

2. Python programming environment

Python programming environment is a convenient and easy tool to use for the completion of the prototype and research works. The main advantages of Python are: simple and clear syntax, a wide access to resources including information, examples, and many ready libraries (including scientific

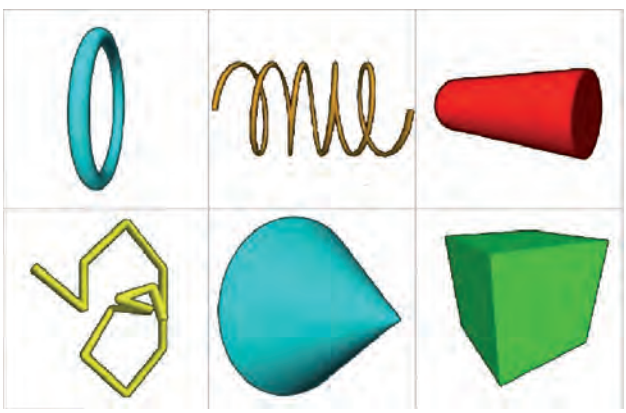

Fig. 2. Exemplary objects found in the Visual library

Rys. 2. Przykładowe obiekty występujące w bibliotece Visual

applications), complete portability between different software and hardware platforms, and the lack of licensing restrictions (open sources). In this paper Visual [6, 8, 9] (for visualization) and pySerial (for serial communication) libraries have been used.

Visual graphics library has many ready objects for immediate use (fig. 2).

In addition, this library allows full automation of the 3D scene management. Even the simplest program using Visual library is equipped with functions of rotating and scaling of the scene. Visual library allows achieving the intended objective by a very simple creation of complicated 3D programs without a specialist programming knowledge.

A sample code – in Python and using Visual library – executing cube rotation is presented below.

```
from visual import *
b = box()
while true:
→ rate(100) # reduce the number of frames to 100 per second
→ b.rotate(angle = pi/100)
```

In Python, instead of brackets (widely used in other programming languages) indents (→) are used.

The second library used in this work is *pySerial* library, which allows convenient use of serial ports (including virtual ports). Establish communication between the computer and external device comes down to a few simple instructions:

```
import serial
# import library
port = serial.Serial(port name, speed)
# opening of the port with predefined speed
```

After connecting, data to external devices can be sent, e.g.:

```
port.write("hello")
```

At the end, the port has to be closed:

```
port.close()
```

3. Model of the robot in the Python programming environment

Model of the robot made in the Python programming environment was built with elementary solids (ready models), which Visual offers. However, there is the possibility of creating your own components, or import from other programs, e.g. CAD, but in this case is necessary to write your own functions.

A sample program code defining the function responsible for modeling servomechanism is presented below.

```
def add_servo():

→ f=frame()
→ sphere(frame=f, color=color.red, radius=1)
→ box(frame=f, length=40, height=40, width=20, pos=(10,-26,0),
color=(.2,.2,.2))
→ box(frame=f, length=54, height=6, width=20, pos=(10,-18,0),
color=(.2,.2,.2))
→ cylinder(frame=f, pos=(0,-6,0), axis=(0,4,0), radius=4)
→ cylinder(frame=f, pos=(0,-4,0), axis=(0,2,0), radius=5,
color=(.2,.2,.2))
→ cylinder(frame=f, pos=(0,-2,0), axis=(0,2,0), radius=11,
color=(.2,.2,.2))
→ return f
```

The modeled components that were used repeatedly are shown in the fig. 3.

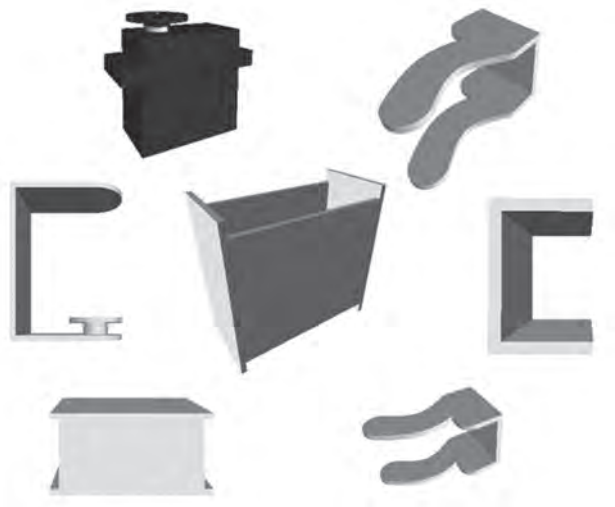

Fig. 3. The basic elements modeled in Python
Rys. 3. Podstawowe elementy zamodelowane z wykorzystaniem środowiska Python

Creating of a robot arm was divided into stages, where first a model of the gripping device (fig. 4) was carried out and placed in the next member, etc. The separate stages

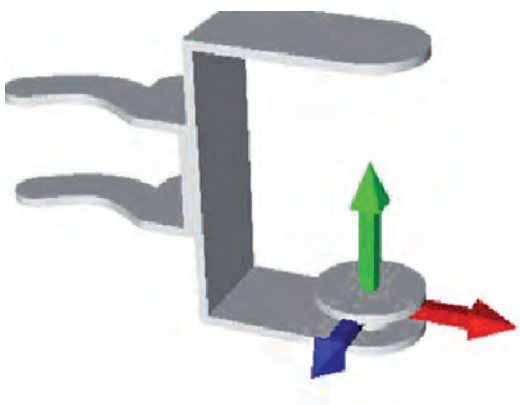

Fig. 4. The gripping device model
Rys. 4. Model urządzenia chwytającego

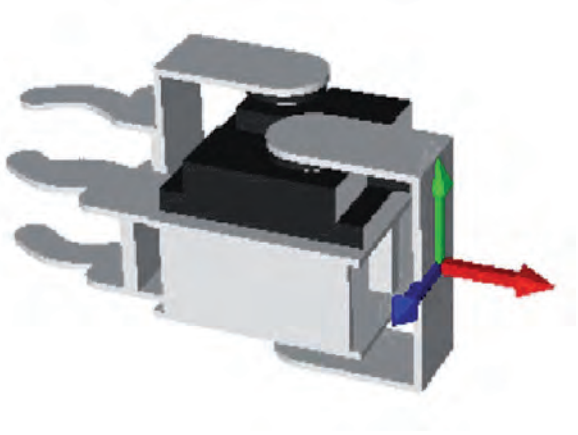

Fig. 5. The first stage of joining elements
Rys. 5. Pierwszy etap łączenia elementów

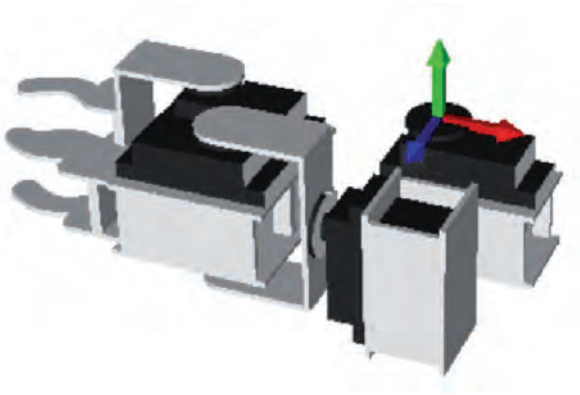

Fig. 8. The fourth stage of joining elements
Rys. 8. Czwarty etap łączenia elementów

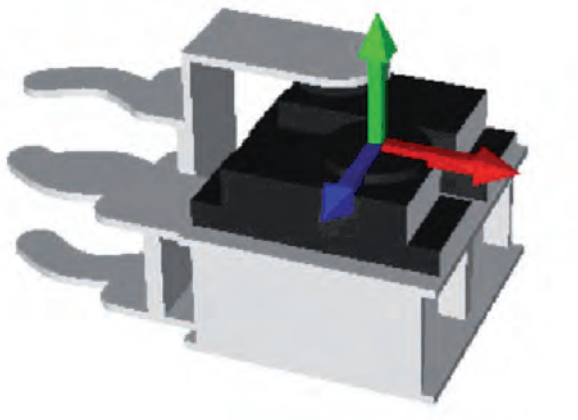

Fig. 6. The second step of joining elements
Rys. 6. Drugi etap łączenia elementów

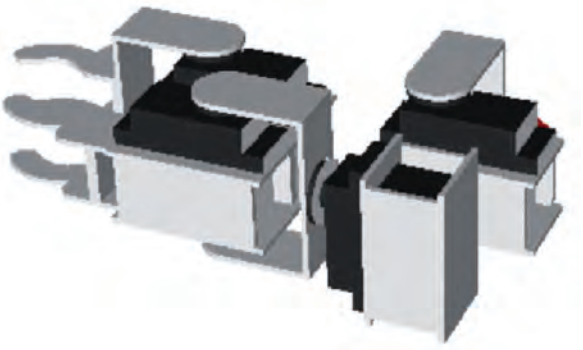

Fig. 9. The fifth step of joining elements
Rys. 9. Piąty etap łączenia elementów

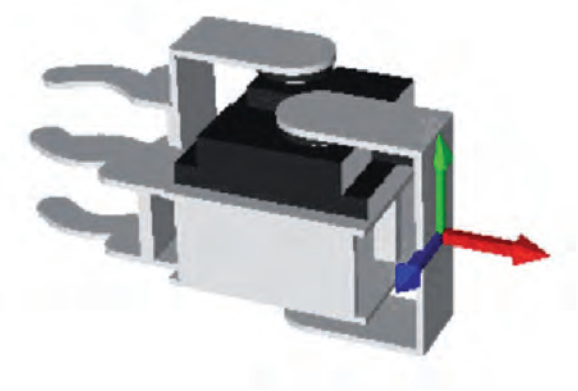

Fig. 7. The third stage of joining elements
Rys. 7. Trzeci etap łączenia elementów

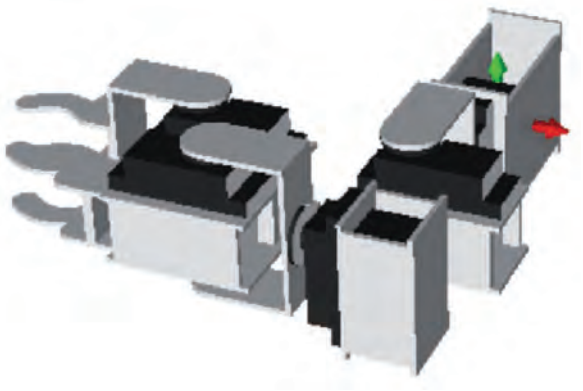

Fig. 10. The last stage of joining elements
Rys. 10. Ostatni etap łączenia elementów

of joining elements in one arm of the robot are illustrated in the fig. 5–10. It should be note that each element has a local coordinate system, which considerably facilitates the control.

When the model is created and during the acting of simulation, components can be rotated by the method rotate(), or can define a new position using the attribute pos. On the "lower level" it is implemented in a typical way for OpenGL [7] – that is matrix operations. In this perspective, the operations of translation, rotation and scaling consist in multiplication of vertex coordinate vector by the corresponding conversion matrix.

A single vertex can be described by a vector v:

$$\mathbf{v} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}.$$

The matrix of rotation around the X axis:

$$\mathbf{M_{RX}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) & 0 \\ 0 & \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

The matrix of rotation around the Y axis:

$$\mathbf{M_{RY}} = \begin{bmatrix} \cos(\alpha) & 0 & \sin(\alpha) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

The matrix of rotation around the Z axis:

$$\mathbf{M_{RZ}} = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Translation matrix:

$$\mathbf{M_T} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Scaling matrix:

$$\mathbf{M_S} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Translation:

$$\mathbf{v'} = \mathbf{M_T} \cdot \mathbf{v}\ .$$

Scaling:

$$\mathbf{v'} = \mathbf{M_S} \cdot \mathbf{v}\ .$$

Rotation around the X axis:

$$\mathbf{v'} = \mathbf{M_{RX}} \cdot \mathbf{v}\ .$$

4. Control of the robot components

Worked out computer program to control the movements of the robot has a few operating modes:

- manual – control with the keyboard and mouse, performed sequences of movements can be stored,
- simulation – executing robot movements on the basis of the loaded sequence,
- automatic.

Three-dimensional user interface accompanies all modes. The program automatically starts up in manual mode, unless a parameter specifying the file with a record of the motion sequence will be given during the start. The change of mode can make at any time of the program. If the robot is not connected to the computer, instead of the automatic mode, the simulation mode will be run.

4.1. Control in manual mode

In order to interact with the user in manual mode, keyboard procedure:

```
if scene.kb.keys:
→ k = scene.kb.getkey()
→ if k=='left':
→ [determined operation]
```

and mouse procedure:

```
m = scene.mouse
if m.clicked:
→ object = m.picked
```

built in library of Visual have been used. These procedures also allowed to significant simplification of the program.

In each loop, the program checks the status of HID devices (*Human Input Devices*). Each keyboard key has an associated own pair of signals, called „scancode. Pressing any key generates an appropriate interrupt signal, which in turn is handled by the subprogram to read and interpret the keyboard buffer. If activity of user is detected, the program checks whether the pressed key is used for controlling the robot. After the identification pressed keys, control functions of individual components position in the space and the velocity change of position are activated.

Operation of mouse comes down to read the clicked object ID, which at a later stage allows you to select the active element of the robot. Information about clicked

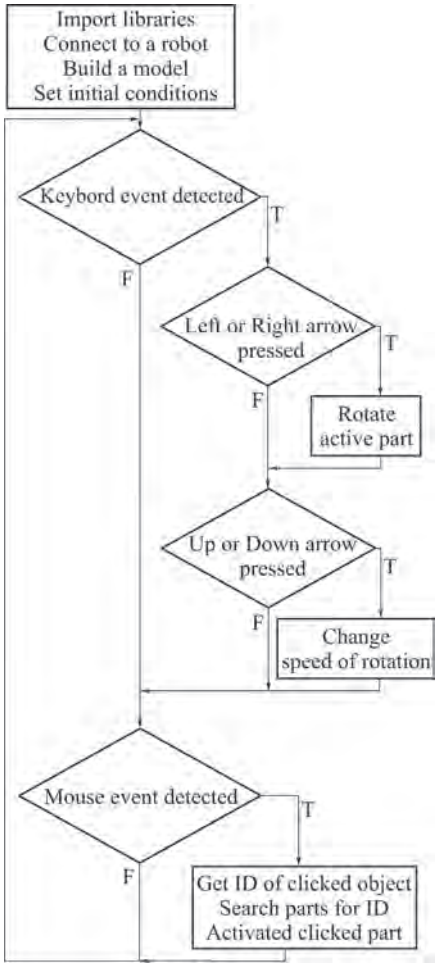

Fig. 11. Robot control algorithm
Rys. 11. Algorytm sterowania robotem

object concerns the lowest level. In order to determine to which elements the clicked object belongs, the recurrent searching of the relation between frames occurs.

A general algorithm of robot arm control is relatively simple (fig. 11):

- if the clicked object is in the frame grouping elements, we make sure that the frame is one of the supported segments of the robot, if not – the same procedure to check for the frame that contains the current element is performed – and so to the moment at which the first condition is not fulfilled, or will be defined in which component the clicked element occurs,
- when they detect on which part of the robot was clicked, activation takes, the color of part changes, and the required variables are setting.

When the user presses on the keyboard the left or right arrows, rotation of active part occurs (angle is set by using the top and bottom arrows). Robot motion is carried out simultaneously in the virtual model, as well as the real object.

The sequence of movements performed by the user can be saved by pressing:

[Enter] – save the last operation,
[Space] – adding a pause for 1 second to the list of operations to be performed,
[S] – save the list of operations in file.

4.2. Control in simulation or automatic mode

If the program is run in simulation or automatic mode (fig. 12), then program gets down to the control mode and executes the commands contained in the loaded script. If the robot is not connected to the computer (simulation mode) is performed only simulation (visualization of robot movements on a computer screen).

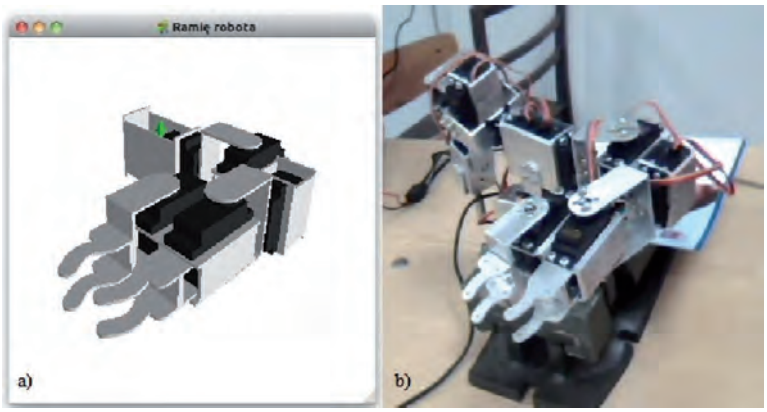

Fig. 12. The modes of robot: a) simulation, b) automatic
Rys. 12. Tryby pracy robota: a) symulacyjny, b) automatyczny

5. Control of servomechanisms

Pololu Servo Controller USB Maestro (fig. 13) was used to control of servomechanisms. With this driver, communication can take place on a serial port TTL (+5 V), which enables for example the Bluetooth adapter or a microcontroller connection, or via the USB cable seen by the computer as a virtual serial port.

Fig. 13. Servo Controller Mini Maestro 12-Channel USB Servo Controller
Rys. 13. Sterownik serwomechanizmów Mini Maestro USB Servo Controller

In order to set the selected servo to the chosen position, connection with the driver must be made, and then control is based on the transfer of three-byte control instruction (fig. 14).

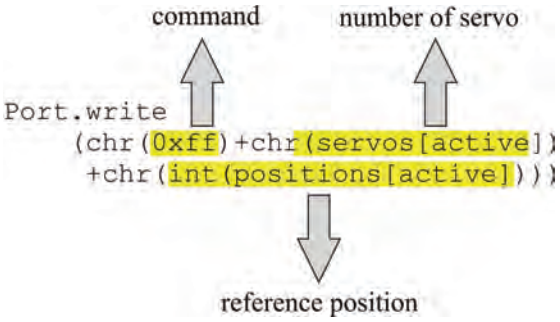

Fig. 14. The control data
Rys. 14. Instrukcja sterująca

6. Conclusion

Despite the fact that all objectives were achieved, developed program in present form is not suitable for controlling the whole walking robot, but only to control his arms. It is difficult to imagine a machine control, consisting of more than twenty servos using only a few buttons. Can thus for instance define the direction in which the robot would have to move, and the rest of the steps can be executed automatically by the walking robot, but it will be still require a lot of effort work on the algorithms that make it will be able to move almost autonomously and control will consisted only on determining some guidelines, and not the positioning of each servo individually.

Acknowledgements

This work was carried out within Scientific Society of Computer Aided Design of Mechatronic Devices and Machines.

Bibliography

1. Klaassen B., Linnemann R., Spenneberg D., Kirchner F., *Biomimetic walking robot SCORPION: Control and modeling*, "Robotics and Autonomous Systems", 41, 2002, 69–76.
2. Ion I.N., Marin A., Curaj A., Vladareanu L., *Design and motion synthesis of modular walking robot MERO*, "Journal of Automation, Mobile Robotics & Intelligent Systems", vol. 2, no. 4, 2008, 25–30 .
3. Zhang X., Zheng H., *Walking up and down hill with a biologically – inspired postural reflex in a quadrupedal robot*, "Autonomous Robots", 25, 2008, 15–24.
4. Bałchanowski J., Gąsieniec P., *Budowa i badania symulacyjne robota kroczącego*, "Acta Mechanica et Automatica", vol. 4, 2010, 9–16.
5. Mrozowski J., Awrejcewicz J., Bamberski P., *Analysis of stability of the human gait*, "Journal of Theoretical and Applied Mechanics", 45, 1, 91–98, Warsaw, 2007.
6. Scherer D., Dubois P., Sherwood B., *VPython: 3D Interactive Scientific Graphics for Students*, "Computing in Science and Engineering", 2000, 82–88.
7. Sterna W., Chodorowski B., *OpenGL I wprowadzenie do programowania gier*, Wydawnictwo NAKOM, 2008.
8. [python.org].
9. [vpython.org].

Zastosowanie środowiska programistycznego VPython do wizualizacji i sterowania robotem

Streszczenie: W pracy przedstawiono sposób wykorzystania środowiska programistycznego VPython (język programowania Python wraz z biblioteką graficzną Visual) do wizualizacji i sterowania ruchami robota. Sposób modelowania oraz sterowania został opracowany dla robota kroczącego, który jest budowany w Instytucie Mechaniki i Podstaw Konstrukcji Maszyn Politechniki Częstochowskiej. Do nawiązania połączenia między komputerem, a modułami robota wykorzystano bibliotekę pySerial. Utworzony program umożliwia pracę w trybie ręcznym (sterowanie za pomocą klawiatury i myszki z jednoczesną obserwacją sekwencji ruchów w czasie rzeczywistym na monitorze komputera), symulacyjnym (realizującym ruchy robota na podstawie wczytanej sekwencji) lub automatycznym. Użytkownik ma możliwość dynamicznego definiowania prędkości oraz położenia pojedynczego serwomechanizmu lub grupy serwomechanizmów.

Słowa kluczowe: robot kroczący, układ sterowania, Python

Maciej Wochal

Student of Mechanical Engineering. Co-founder member of Scientific Circle of Computer Aided Design of Mechatronic Devices and Machines. Scientific interests: programming, navigation and construction of mobile robots, digital control systems.

e-mail: maciekwochal@o2.pl

Dawid Cekus, PhD

Adjunct professor at the Institute of Mechanics and Machine Design Foundation of Technical University of Czestochowa. Tutor and co-founder member of Scientific Circle of Computer Aided Design of Mechatronic Devices and Machines. Scientific interests: studying the dynamics of mechanical systems, the use of genetic algorithms in the modeling of machines and their components.

e-mail: cekus@imipkm.pcz.pl

Pawel Warys, PhD

Adjunct professor at the Institute of Mechanics and Machine Design Foundation of Technical University of Czestochowa. Author and co-author of several publications from this field in national journals. Scientific interests: static and dynamics research of mechanical systems (especially the forest crane and its subassemblies).

e-mail: warys@imipkm.pcz.pl