TALLINNA TEHNIKAÜLIKOOL
MEHAANIKATEADUSKOND

Department of Mechatronics

Chair of Mechatronics Systems

MHK70LT

Islam Bzhikhatlov

158877MAHM

BIPEDAL ROBOT WITH SIMULATION MODEL FOR TEACHING ROBOTICS

Master's Thesis

The author applies for
the academic degree
Master of Science in Engineering

Tallinn
2016

TALLINNA TEHNIKAÜLIKOOL
MEHAANIKATEADUSKOND

Mehhatroonikainstituut

Mehhatroonikasüsteemide õppetool

MHK70LT

Islam Bzhikhatlov

158877MAHM

KAHEJALGNE SIMULATSIOONIMUDELIGA ROBOT ROBOOTIKA ÕPETAMISEKS

MSc Lõputöö

Autor taotleb
tehnikateaduste magistri
akadeemilist kraadi

Tallinn
2016

(The reverse of the title page)

AUTHOR'S DECLARATION

I hereby declare that this thesis is the result of my independent work.
On the basis of materials not previously applied for an academic degree.
All materials used in the work of other authors are provided with corresponding references.

The work was completed .... guidance

“.......”....................201….a.

The author

.... signature

The work meets the requirements for a master's work.

“.......”....................201….a.

Supervisor

.... signature

Permit to defense

.... curriculum defense superior

“.......”...................201… a.

.... signature

TUT Department of Mechatronics
Chair of Mechatronics Systems

MASTER'S THESIS SHEET OF TASK'S

Year 2016... semester …2……

Student: Islam Bzhikhatlov 158877 MAHM
Curricula: Mechanics
Spetsiality: Mechatronics
Supervisor: Research Scientist, Maido Hiiemaa

MASTER'S THESIS TOPIC:
(in English) BIPEDAL ROBOT WITH SIMULATION MODEL FOR TEACHING ROBOTICS
(in Estonian) KAHEJALGNE SIMULATSIOONIMUDELIGA ROBOT ROBOOTIKA ÕPETAMISEKS

Thesis tasks to be completed and the timetable:

Nr	Description of tasks	Timetable
1.	**Development of the physical bipedal robot**	**10.12.2015**
2.	**Creating 3D model of the robot**	**20.12.2015**
3.	**Creating behavioral model of the robot (physical behavior + simplified servo models)**	**20.02.2016**
4.	**Creating interface(s) between a PC and the robot**	**25.03.2016**
5.	**Testing (comparison of the virtual and the physical robot)**	**25.04.2016**

Solved engineering and economic problems:
Development of a bipedal robot with visualization to be used as a teaching tool in robotics.
The robot must use low-cost R/C servos and 3D printable parts instead of industry-grade components to bring the cost down and to emphasize the imperfections, which must be modeled and analyzed by students.

Additional comments and requirements: ….…………………………………..........................

Language: English

Application is filed not later than 16.05.2016
Deadline for submitting the theses 20.05.2016

Student Islam Bzhikhatlov /signature/ ……….......... date ………
Supervisor Maido Hiiemaa /signature/ ……….......... date………

Confidentiality requirements and other conditions of the company are formulated as a company official signed letter

CONTENTS

Introduction 7

1. REQIREMENTS AND GOALS 9

2. DEVELOPMENT OF THE PHYSICAL BIPED ROBOT 11

2.1 Principles of locomotion 11

2.2 Kinematics 13

2.3 Leg joints location 14

2.4 Leg dimension 14

2.5 Weight and center of gravity 15

2.6 Approximately calculation of servos 16

3. CREATING 3D MODEL OF THE BIPED ROBOT 18

3.1 Modeling software 18

3.2 Modelling of biped robot using CAD software 19

3.3 Simulation software 22

4. CREATING BEHAVIORAL MODEL OF THE ROBOR 24

4.1 Importing data of model to V-REP and creating of scene 24

4.2 Setting options and properties of shapes and joints 27

4.3 Graphs in V-REP 29

4.4 Sensors in V-REP 31

4.5 Simulation features 31

5. CREATING INTERFACE(S) BETWEEN A PC AND THE ROBOT 34

5.1 Data communication 34

5.2 Servo controller and control scheme 35

5.3 Modifying of electronic components 37

6. TESTING (COMPARISON OF THE VIRTUAL AND THE PHYSICAL ROBOT) 38

6.1 «Pre targeting» mode 38

6.2 «Real targeting» mode 41

7. CONCLUSION 44

7.1 Collisions and problems 44

7.2 Future work 45

REFERENCES 48

APPENDIX A 50

APPENDIX B 57

APPENDIX C 66

Introduction

Two legged robot locomotion mechanisms are very popular and complex field of science. Engineers can’t make robot legs so as it exists in biological systems, which are very successful and don’t have any problems to walk, jump and run [1]. However, there are a large number of two-legged robots that work on simpler principles than human legs. These simplified principles [2] are still hard to study for engineering students. Problems in the study of biped robots have existed for a long time and solution to this problem is complicated due to high price of existing biped robots. The high cost of such robots rarely allows universities to provide all students with the necessary equipment, which is necessary for completely understanding principles of creating and improving of them. Even if the organization has enough money to buy this robot, during the study there might be cases, when robot is breaking. It’s not very good for university and teachers sets limitation of using robots in learning process for students and disallows to experimentalise, and even, in some cases disallow to touch. This situation does not promote making studying the difficult biped robots more easy. Sometimes, students can be allowed study robots so as they need, but here there is another problem. Most parts of biped robots have protected case, which hidden the components, joints and very important points.

In most cases students don't get knowledge about simulations and visualizations of robotics systems or face with the complexity of simulation and visualization of robots in the process of studying. Another problem is developing of algorithms and programs [3] for biped robots and testing. In developing process of algorithms it is very important to have visualization, which saves a lot of time and gives most useful information about errors.

Price of advanced biped robots is quite high. Cheap biped robots are available, but they cannot be improved, modified, simulated and tested.

Possibility of simulation allows students programming devices so as like him, studying results, positives and negatives of changing approximately even before uploading the program to device.

In our days, it is difficult to imagine a robotic system without the system of simulation and visualization. For example, large companies such as ABB Robotics and KUKA robotics use their own simulation systems. However, designing such systems manually for educational robots is not possible due to their complexity. There are different virtualization systems robots that have the ability to connect to any of the robots, but you need to create a 3D model, scenario simulations, programm it and provide it with correct exchange of data.

Important point of developing of biped robots is existence of some ready platform, which gives the base information about main points and can be used as a base of future robot.

1. REQIREMENTS AND GOALS

Requirements and goals are determined by the needs of simplified and useful solution for teaching capabilities of simulation and visualization. There are many biped robots which can be used as a studying platform. Features of specifications don’t allow modifying and improving those robots, or complexity disallows studying main principles that also makes impossible to modify it by yourself.

For studying platform there were determined the next main requirements: low-cost, replaceable parts and ability to change basic parameters. Rapid prototyping technology allows making the parts so as we want and allow decreasing cost of components to the maximum. Even for cases when some parts are broken, we can replace it easily, without limitation for studying process and quite fast.

Another positive factor for studying process is getting experience to rapid prototyping on practice. As mentioned, there are a lot of ready biped platforms, some of them for low cost. But all of them which are not printable can't be modified and improved, therefore it is a bad option for studying.

A lot of attention is paid to the accuracy of manufacture parts in industrial robots. For studying tools it is not required since it all are stuffing in other subjects or unimportant for beginners.

There are several projects, which can be used as a teaching platform, but the majority of them is quite complex and can cause some misunderstanding for students. For example, "Poppy" robot [4] (figure 1.1 at left) was made in France as open source project. This solution has printed parts and more or less accessible electronic components. Unfortunately this solution is too difficult for students, which have not experience in robotics and mechatronics due to using of theoretical aspects too much more.

There are other biped robots with printed parts, which were created by enthusiasts and have not mathematical base and software. The most interesting of them, in my opinion, is “HaRo” robot [5] (figure 1.1 at right) which has 10 DOF (Degree of freedom).

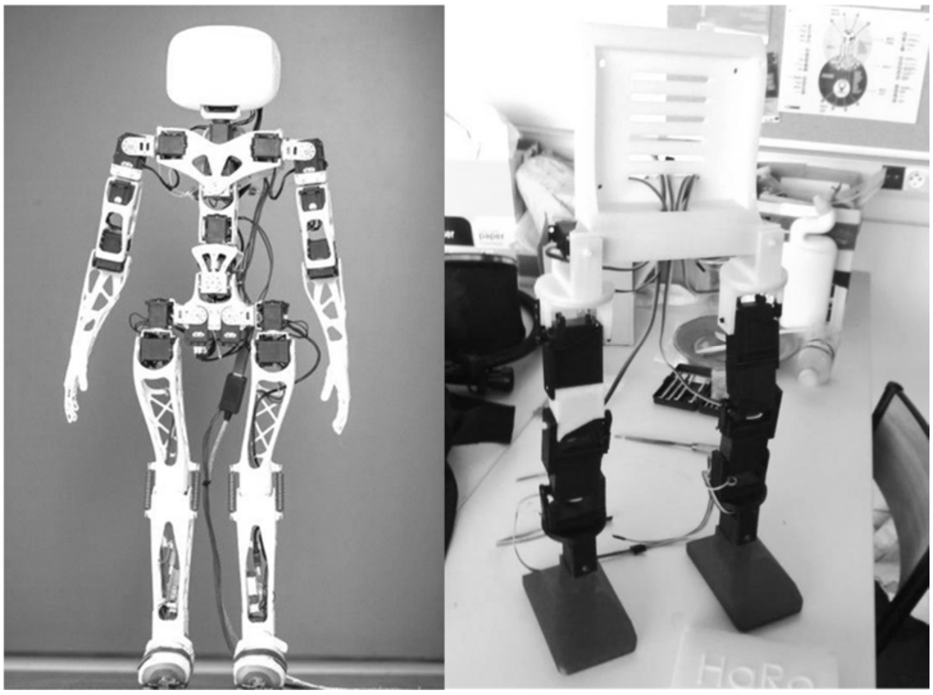


Figure 1.1 – “Poppy” robot (at left) and “HaRo” robot (at right)

Usage of the low-cost R/C servos also can allow making biped robot simpler and cheaper, but some problems with current positions of servos appear. The low-cost servos do not allow receiving current position since the logics were realized inside of servo. Current position is very important for control algorithm. There is another problem concerning reliability and stability of construction. Typical problems of two legged robots can be solved in difference degree that dives some freedom for development. In some cases it depends on the complexity of task for biped robot.

Problems with simulation and visualization are also quite important. Some software exists in this field but it takes too much time for studying and simple manuals for biped robots are not available. Also that software has to be tested.

After analysis the next main requirements were formulated: the robot must use low-cost R/C servos and 3D printable parts instead of industry-grade components. Also robot must have imperfections (which must be modeled and analyzed by students) and simulation.

I tried to solve all these questions in my work using available tools and ready solutions. Also in my work I tried to stick to the accepted rules and laws of creation and control of bipedal robots. My work is not going to be the best solution for bipedal platform; this platform is going to allow modifying, improving and developing the bipedal robots for studying robotics.

2. DEVELOPMENT OF THE PHYSICAL BIPED ROBOT

Development of biped robot was started from locomotion since it is the main function of biped robot and features of biped robot based on principles of locomotion.

2.1 Principles of locomotion

For start all main approaches and principles of creating biped robots were found. Unlike biped robots, wheeled locomotion is very popular in our day. Walking principle is like to wheeled locomotion if the step size decrease [6].

For each locomotion principle (it doesn't matter if it is wheeled, legged or another) there are three basic issues: stability, the characteristics of ground contact and the type of environment, as described in [7].

The main principle, which holds most of the modern biped robots, is minimization of fluctuations in position of the robot gravity center. Linear motion position of the gravity center with a constant speed or slowly changing speed ensures the successful robot locomotion.

The dynamic and static stability is the main task, which has to be solved at the beginning. Static stability means that in any time biped robot have to be stable and don't have to fall. For bipedal robot it can be supposed that it is not possible because the robot has only two legs and one leg is not allow solving this task during the motion. Due to described above reasons the problem of stability is solving at another level.

The majority of walking robots has a big foot for solving this issue and has a large contact surface. Next is necessary new term which can be determine as support polygon, as describe in [6] only robots with four or more legs can be static stable. In some cases, support polygon is not equal to contact surface. Increasing the support polygon and decreasing the contact surface into needed limitation doing the biped robot more stable. It doesn't mean that bipedal robot is completely static stable, but it means that biped robot can be more stable based on principles similar to static stable.

Next important think is dynamic stability. General algorithm to solve the problem of dynamic stability for bipedal robots still does not exist. For the most of biped robots approaches based on the zero moment point (ZMP) [8] are used. ZMP specifies the point with relation to which dynamic reaction force at the contact of the foot with ground doesn't produce moments in the horizontal direction [9]. Zero moment is necessary for biped walking, because various factor such as join moving or the constant changing of the Center of Mass [10] can unstable the robot. It is only one of reasons to calculate the trajectory of the Center of Mass before making new step, because it depends on ZMP. The position of ZMP is affected by mass and inertia of the robot's hull, since its motion usually requires large ankle torques to maintain a satisfactory dynamical postural stability. Decreasing the torque in ankle can solve this problem.

When the robot's foot contacts the ground it is influenced by a reaction from the ground called the floor reaction force. An ideal walking pattern is created by the computer and the robot's joints are moved accordingly. The total inertial force of the ideal walking pattern is called the target total inertial force, and the ZMP of the ideal walking pattern is the target ZMP.

The ZMP is the point where the robot has to base on to keep its balance. When the robot should move forward it has first to compute the ZMP and after that it has to step the appropriate leg exactly to the computed position. The Zero Moment Point1 (ZMP) is often described in robotics as the point on the ground where all momentums are equal to zero. The ZMP can be computed with equation (1) and (2) (as described in [11]):

$$\mathrm{x_{zmp}} = \frac{\sum_i \mathrm{m_i(z+g)x_i} - \sum_i \mathrm{m_i x z_i} - \sum_i \mathrm{I_{iy}}\theta_{\mathrm{iy}}}{\sum_i \mathrm{m_i(z+g)}}$$

$$\mathrm{y_{zmp}} = \frac{\sum_i \mathrm{m_i(z+g)y_i} - \sum_i \mathrm{m_i y z_i} - \sum_i \mathrm{I_{ix}}\theta_{\mathrm{ix}}}{\sum_i \mathrm{m_i(z+g)}}$$

Where $(x_{zmp}, y_{zmp}, 0)$ are the ZMP coordinates in the Cartesian coordinate system, (x_i, y_i, z_i) is the mass center of the link *i*, is the mass of the link *I*, and g is the gravitational acceleration. I_x and I_y are the inertia moment components, θ_{ix} and θ_{iy} are the angular velocity around the axes x and y (toke as a point from the mass center of the link i).

One more problem of biped robots is Foot-Rotation Indicator (FRI) Point, i.e. foot rotation in biped robots during the single-support phase. The foot-rotation indicator (FRI) point is a point on the foot/ground-contact surface where the net ground-reaction force would have to act to keep the foot stationary as described in [12].

Thus, for achieving stable walking as described in [13] is necessary to

- Target ZMP Control
- Floor Reaction Control
- Foot Planting Location Control

[14] presents quite interesting research about aspects of redundant actuation when the robot is a closed kinematic chain. As were described, process of walking is quite complex and can‘t be provided in fully. Described principles must be taken into account for next developing of biped robot.

2.2 Kinematics

The number of freedom’s degrees was one of difficult and important aspects of developing the robots. Biped robot with minimum DOF with quite big functionality was described in [15] [15]. Each leg has five DOF. During the choice the next issues were taken into account: how much more complicated the robot when you add more DOF and how strong the restriction is in reducing the number of DOF. Five DOF is optimal for studying platform because it allows saving the most of functionality and remove insignificant for studying at the beginning.

The total DOF of the mechanism was calculated as described in [16]. Results were presented belong: $DOF = 6 \cdot n - 5 \cdot P_5 - 4 \cdot P_4 = 6 \cdot 6 - 5 \cdot 2 - 4 \cdot 4 = 10$

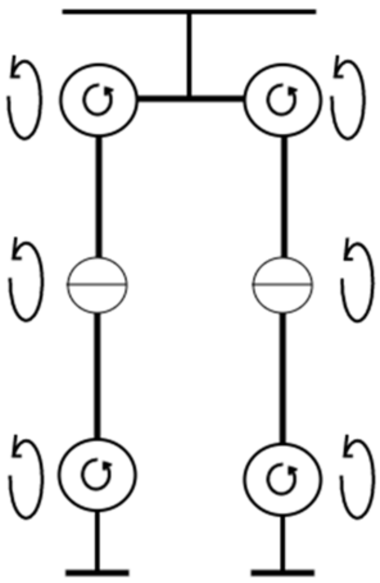


Figure 2.2 – Simplified kinematic scheme

Contact surface was taken as a flat surface with big hole. It allowed making robot more statically stable. The center of mass was located depending on servos location, since servos are more heavy compared to the other parts.

2.3 Leg joints location

So as our robot is biped and considered as more or less like to humanoid robot, the lower part of human skeleton was used for reference when locating the leg joints. Range of joint movement during walking was based on the capabilities of the servo.

For first experimentation approach of maximum simplifying biped robot locomotion was taken, but this concept acts contrary to the main principles, which described above. That concept had eight DOF. Designing from easily to more difficult is the only one reason for creating simplified concept but for this case 8 DOF concept was not able to be like a humanoid even as simplified to the maximum.

New concept was created depending on theoretical part. New concept has 10 joints (DOF). The most important of them was modifying of foots. It was very important to decrease the foot contact surface and increase support polygon [6], it made the robot more sustainable.

2.4 Leg dimension

The ability to change the size of one important robot elements without additional changes to other parts was incorporated in principle of developed model. It ensures the modularity of biped robot. In process of learning this possibility allows the students to make robots of different parameters and to study their behavior, characteristics, and influence on behavior, which depends on size parameters.

Universal mounts also allows you to change not only their length, but also width and length. In addition, it is possible to change design features, to make holes for attaching additional devices. However, there are limitations, when you increase the size of some parts by more than 20%, it is also recommended to recalculate parameters.

Leg dimension is most of important point in biped robots. As were described in principle of locomotion, it is important for static stability of robot. Also, foot prints (contact surface) are very important for static stability walking. Statically stable walking keeps the projection of the center of mass inside the support area (polygon). (More details were determined in [17]).

For better understanding and possible comparison, the following is table of Human Proportion Data (Table 2.1). Data were taken for the 50th percentage of United States males aged from 18 to 70 [18] and can be used for correlation. As can be seen from table 2.1 all dimensions are approximately to the same extent reduced except the foot breadth.

There are several reasons for this. Firstly, it allows making sustainability of biped much more. Secondary, it lets increase the radius of movement of the center of gravity in the horizontal plane. Also the needed breadth was can be calculated approximately with a large margin.

Dimension	Value for human (mm)	value for biped robot (mm)
Foot length	245	95
Foot breadth	95	80
Popliteal height	440	95
Knee height	545	155
Hip height	920	260
Hip breadth	360	125

Table 2.1 – Human Proportion Data vs. biped robot proportion

The width of the hip was relatively increased to make possible the implementation of different configuration principles of the thigh. One of interesting principles of hip designing presented in [19 pp. 9-11]. The size and proportions of the biped robot have a strong dependence on the method of construction and design, placement of joints and motors. Therefore, at the stage of designing, it is recommended conducting additional calculations in the software packages of finite-element modeling. This will allow finding the weak points in the design.

2.5 Weight and center of gravity

One of the unique features of the printed materials is the low weight. Usually for printing acrylonitrile butadiene styrene plastic (ABS) or Polylactic acid (PLA) are used. Those materials have a rather low weight while remaining fairly durable material. PLA has a feature shrinkage (reduced in size a little bit).

A biped robot has to maintain control over its center of mass (CM) and possible force disturbances during walk. Used approach does not take into account the weight of printed parts, because the using material is synthetic polymer. But it is not about servos, each servo have a weight equal 50 grams.

Center of gravity is calculated using Solid Works Motion software for all important positions. It allows determining the limitations in horizontal directions for movement. Here only allowable movement of robot parts that determined the position of CM was calculated. Possible dynamic parameters that can influence the position of CM were not taken into account, because it assumes that the joints of robot will move quite slowly.

2.6 Approximately calculation of servos

Any movement mechanism is requiring using force source. Usually biped robot use pneumatics, electric motors or both of them together. We have to consider the approximately calculation of minimal required parameters for biped robot. For our case the most appropriate option is servos since it depends on dimensions of robot and available power supply.

To determine the required minimum torque of the servo we take the case where the shaft operates the servo maximum torque – it is bottom servos. Concerning to the left lower servo will conduct the calculations.

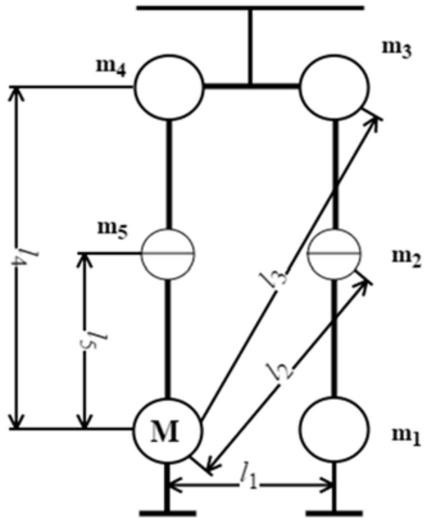


Figure 2.2 – Illustration of methods of calculating the servo

Simplified maximum load can be calculated as:

$$M_{lo} = M_1 + M_2 + M_3 + M_4 + M_5 = F_1 \cdot l_1 + F_2 \cdot l_2 + F_3 \cdot l_3 + F_4 \cdot l_4 + F_5 \cdot l_5 = m_1 \cdot g \cdot l_1 + m_2 \cdot g \cdot l_2 + m_3 \cdot g \cdot l_3 + m_4 \cdot g \cdot l_4 + m_5 \cdot g \cdot l_5$$

There g=9.81 m/s^2 - Gravitational acceleration. $m_i -$ mass of each servo (for our case it was taken around 50 grams), $l_i -$ distance to servo ("shoulder").

$$M_{lo} \approx 0{,}05 \cdot 9.81 \cdot 0.105 + 0{,}05 \cdot 9.81 \cdot 0.159 + 0{,}05 \cdot 9.81 \cdot 0.248 + 0{,}05 \cdot 9.81 \cdot 0.225 + 0{,}05 \cdot 9.81 \cdot 0.12 \approx 0.42\ (N \cdot m)$$

M_{lo} is the most simplified load value on shaft of servo and in some cases load can be more than this value. But we can say that the load can't be more then $2 \times M_{lo}$ this value we will take as a required torque for our servo.

$$M_{max} = 2 \times M_{lo} = 2 \cdot 0.42 \approx 0.84\ (N \cdot m)$$

Also for our case one more requirement exists - Gear Material mast metal. From catalog of servos [20] was found low-cost RC servo Tower Pro MG-995 (shown on figure 2.3).

Servo MG-995 [21] has next specifications:

- Weight: 55 g
- Dimension: 40.7 x 19.7 x 42.9 mm approx.
- Stall torque: 8.5 kgf·cm (4.8 V), 10 kgf·cm (6 V)
- Operating speed: 0.2 s/60º (4.8 V), 0.16 s/60º (6 V)
- Operating voltage: 4.8 V a 7.2 V
- Dead band width: 5 µs
- Stable and shock proof double ball bearing design
- Temperature range: 0 ºC – 55 ºC

So as $10\ \text{kgf} \cdot \text{cm} \approx 0.98\ \text{N} \cdot \text{m}$ this servo is suitable for our case.

Figure 2.3 – Servo Tower Pro MG995

3. CREATING 3D MODEL OF THE BIPED ROBOT

3D model of biped robot is created depending on previous parts of work. In this stage of development is primarily determining joint lengths, their design features, possibility of installation of electronic component and their interaction. So as will be used electronic components which were determined before and which has to be visible, should be it all taken into account. Also, should be calculated strength (minimum and maximum stability). In addition, the design should be resistant on two supports or on one.

To determine the all parameters, requirements for biped robot should be taken into account. Despite the fact that basis of robot kinematics was took from human skeleton, the ratio of the lengths of the links need to be modified due to the difference in the location of servo motors and differences in the operation principle of biped robot. Following parameters is depends on joints length: target ZMP, Floor reaction, the time required for one step and all parameters required for his calculation, center of mass, and all other physical parameters of robot.

3.1 Modeling software

There are various simulation programs, which are well suited for tasks of modelling. Such parameters as accessibility in universities and the prevalence of the use (presence of knowledge and skills to use) were considered as a most important. All files must be saved in the STL format for 3D printing and usually this format also used to importing data into another software. Therefore only software with described option can be considered. But such programs are a huge number and this feature is supported even in programs that are not designed to work in mechanical engineering.

After software analysis, highlighted three software's, each of them is maximum suitable for the described task and strongly recommended to use only these software's. Following programs: Solid Edge, Solid Works, and Inventor are pretty popular for mechanical engineering. Mentioned software's are very user friendly. Usually, this software (or one of them) is studied in higher educational institutions at the beginning of studying. But another CAD (Computer-aided design) or CAE (Computer-aided engineering) systems like mentioned software's also can be used.

SolidWorks was determined as modeling software for this type of tasks: user friendly so as project is going to be oriented to more beginners.

3.2 Modelling of biped robot using CAD software

Problem with fastening of servo was revealed at the beginning of modelling. Connection of servo shaft with other parts was not determined also. It was decided to fixate the servos through another small element so that the axes of servos were perpendicular (figure 2.1). This allowed securely fasten the servos with each other, also provided with the necessary holes for fixing another elements of biped robot.

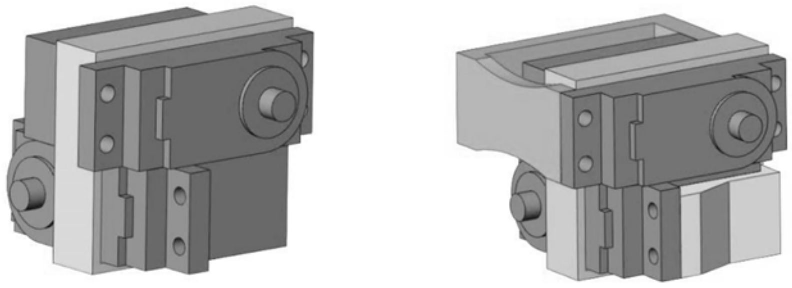


Figure 2.1 – Fasten the servos was modeled in SolidWorks software

This method of fastening allows establishing the upper and lower servo using the same elements. However, one of the binding around the body is not enough for accurate fastening. For this reason were used the additional elements for each servo as shown on figure 2.1 (right side). Those additional elements for fastening are same to the upper and lower, left and right side.

Much attention was given to the knee joint during the process of modeling. An important condition that ensures the correct functioning of this connection is a method of fixation and the location of servo. As most suitable approach was determine fixation on bottom part of leg as shown on figure 2.2 (left side). Fixation was provided by installing a servo in a groove, which is perfectly suited to the dimensions of the front (shown on figure 2.2).

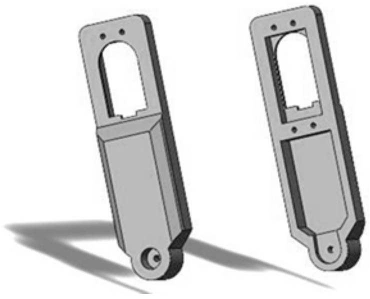


Figure 2.2 – bottom part of legs

Also the servos are mounted to the one part of leg by four screws so that they are securely fastened. The second part of dogs is connected to the shaft of the servo. The second part of dogs is connected to the shaft of servo by special "hat", which is content of servo kit (shown on figure 2.3 at right). This item attaches directly to the servo shaft by one screw. Also 'hat' is fasten to second part of leg by two screws.

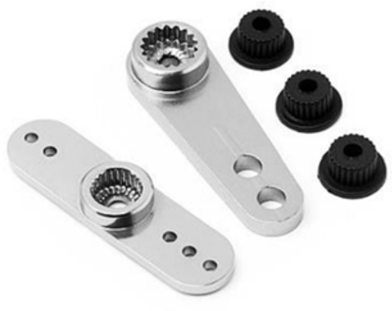


Figure 2.3 – Fastening element between the servo and the upper part of leg

As mentioned above, the fastening element was attaching to the second part of leg by two screws and also special slot was made for this element (shown on figure 2.4) since connection has to be attached straight and constantly ensure rigid connection.

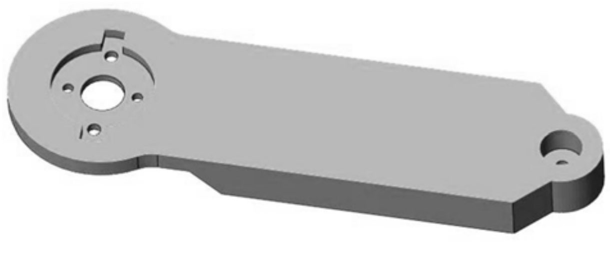


Figure 2.4 – 3D model of leg second part

"Ankle" is another important element in the design. This part is one of the most complex and the studied parts of biped robot. Many scientists conduct research and identify features of their development. For our case we consider a simple concept with flat contact surface.

The first model, which was modeled, is presented in figure 2.5. This approach had one connection points with servo shaft and was not unsought for providing the functionality.

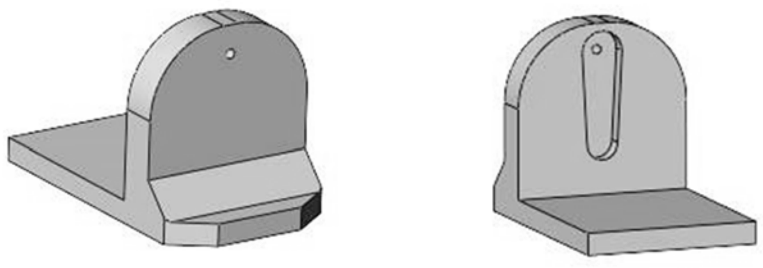


Figure 2.5 – 3D models of ankles as a first approach

This model was improved by creation addition connection point like a bearing, which provided needed loads and freely rotation. As shown on figure 2.6 (right side), this approach allows using the additional fastener by making only one hole for shaft.

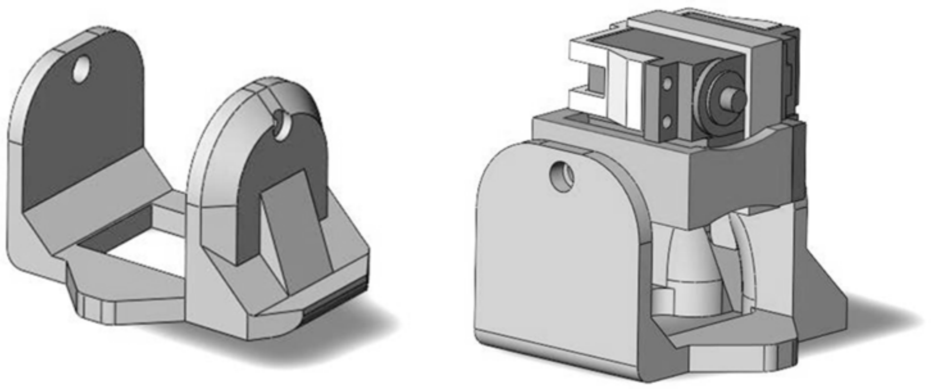


Figure 2.6 – 3D models of ankles as a final design

Also contact surface was modified so that the area of contact surfaces was decreased and support of foot was increased. Design features of the contact surfaces are very complex and knowledge intensive task, so in this work it was not given much attention. However, the data presented is sufficient for forming a view on the basic criteria of its development.

3D models were assembled also in Solidworks software. It is necessary to be made correctly for next importing into simulation software. Collision of 3D models is main field for attention during the assembling. The problem is that SolidWorks allows to collect parts so that their surfaces can intersect; however, in simulation it is not permissible. This important aspect has to be taken into account.

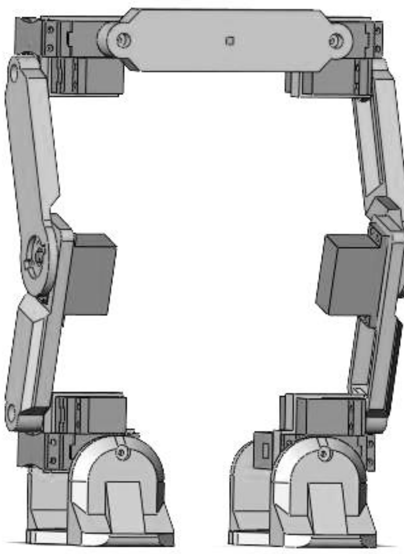


Figure 2.5 – Biped robot was modeled as assembled using Solidwoks software

3.3 Simulation software

The choice of software for the simulation differs significantly from the software selection for modelling. A well-designed simulator makes it possible to rapidly test algorithms, design robots, and perform regression testing using realistic scenarios. There are software packets for modeling not so much as for simulation due to of its complexity. The problem of analysis of existing solutions and their capabilities also is a difficult task.

Basic requirements: accessibility (free), friendly user interface and wide opportunities. After a detailed analysis two software solutions were selected. They have convenient programmatic and graphical interfaces: Virtual Robotics Experimentation Platform (V-REP) [22] and Gazebo [23]. Both of them are free accessible. Gazebo is open source project and free for using. V-REP has free educational license. So as developing biped robot for studying process, education license is enough for our case. As mentioned on official web site, Gazebo is powerful tool for robotic simulation, but V-REV has almost same functionality and friendlier user interface, which making communication more easy and effective.

The main advantage of V-REP is the flexibility and expandability, integration and communication with other software solutions.

The V-REP was chosen for simulation. V-REP is virtual simulation platform made by Coppelia Robotics. Mentioned software is integrated development environment, is based on distributed control architecture: each object/model can be individually controlled via an embedded script, a plugin, a ROS node, a remote API client, or a custom solution. This makes V-REP very versatile and ideal for multi-robot applications. Controllers can be written in C/C++, Python, Java, Lua, Matlab, Octave or Urbi.

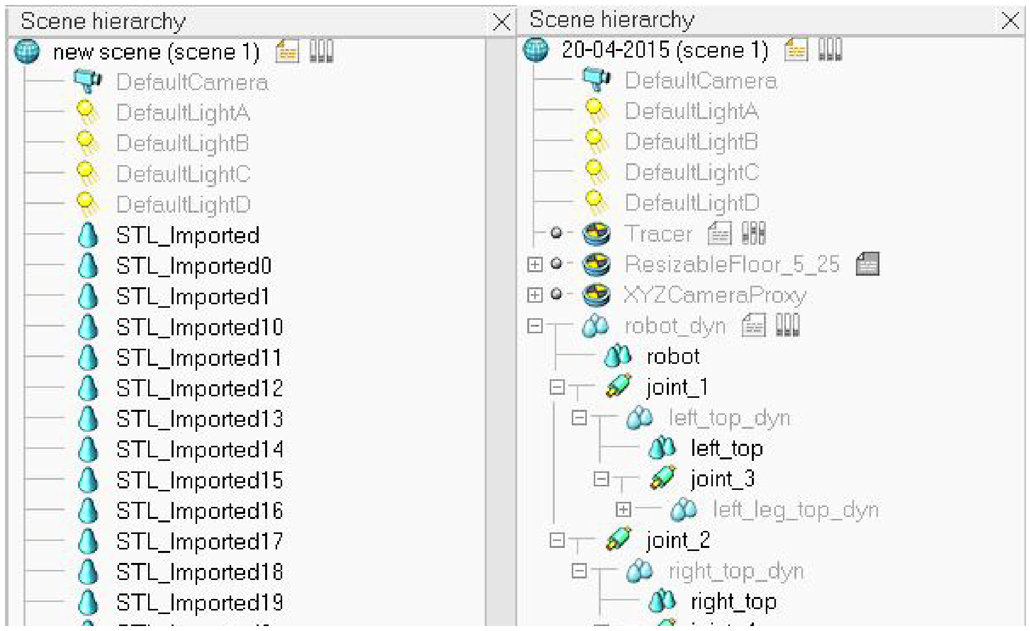
V-REP is using for fast algorithm development, factory automation simulations, fast prototyping and verification, robotics related education, remote monitoring, safety double-checking, etc.

4. CREATING BEHAVIORAL MODEL OF THE ROBOR

Creating of behavioral model is important part of simulation which allows to make simulation more realistic. Behavioral model of a robot depends on the control algorithm and its capabilities, which were laid out in the design of biped robot.

4.1 Importing data of model to V-REP and creating of scene

V-REP allows importing of models in STL (STereoLithography) format. Since initially models were created in Solid Works software, all elements were saved in STL format, after that were imported into V-REP. Since initially models were created in Solid Works software, all elements were saved in STL format, after that were imported into V-REP. After importing of components the model hierarchy of scene contents new positions with standard names (i.e. all objects is used in scene). Structure of used elements is presented as a hierarchy tree and all elements should be located correctly relatively to each other for correct simulation. Scene objects have to be built in an appropriately hierarchy (was shown on figure 4.1 at left). Each element with default name has to be renamed for more complete understanding of contents. Double-clicking on name allows editing it.



```
Scene hierarchy
new scene (scene 1)
    DefaultCamera
    DefaultLightA
    DefaultLightB
    DefaultLightC
    DefaultLightD
    STL_Imported
    STL_Imported0
    STL_Imported1
    STL_Imported10
    STL_Imported11
    STL_Imported12
    STL_Imported13
    STL_Imported14
    STL_Imported15
    STL_Imported16
    STL_Imported17
    STL_Imported18
    STL_Imported19
```

```
Scene hierarchy
20-04-2015 (scene 1)
    DefaultCamera
    DefaultLightA
    DefaultLightB
    DefaultLightC
    DefaultLightD
    Tracer
  ⊞ ResizableFloor_5_25
  ⊞ XYZCameraProxy
  ⊟ robot_dyn
      robot
    ⊟ joint_1
      ⊟ left_top_dyn
          left_top
        ⊟ joint_3
          ⊞ left_leg_top_dyn
    ⊟ joint_2
      ⊟ right_top_dyn
          right_top
```

Figure 4.1 – Default scene hierarchy (left) and Structured scene hierarchy (right)

Objects in the scene hierarchy should drag and drop into another object, in order to create a parent-child relationship. A model was defined by attaching all objects that logically belong to the model as a base object. At the result were got correct scene hierarchy as shown in figure 4.1 (at right).

All joints were defined as appropriate connection type. For this case all joints is "Revolute". Joints also have to be located appropriately in scene hierarchy. In addition joints have to be positioned correctly in scene and have appropriately coordinates (set correctly coordinates in scenario).

Position of joints in V-REP requires some additional actions. For correct results the positioning of joints should be select relation to shaft, i.e. shaft of servos for this case. For this task was creating new scenario, selected and copied all servos to new scenario. Next actions have to be done only in new scenario. Here "convex decompression" shapes should be added. A “convex decompression” is a convex mesh which optimized for dynamics collision response calculation. "Convex decompression of selection" was added to scenario on next step by selecting the servo, clicking right button of mouse and choosing "add". All default parameters in modal window are correct except "Maximum Concavity" which has to be equal 20 mm. After this operation in the scene hierarchy will appears new element. The visibility of this element should be shifted to another layer and switched to the layer with convex element by using command "Tools" and "layers". In new small window layers can be switched. On figure 4.2 was shown how the convex shape looks. Facing to problems with visibility layers, should be taken into account that V-REP has Help file.

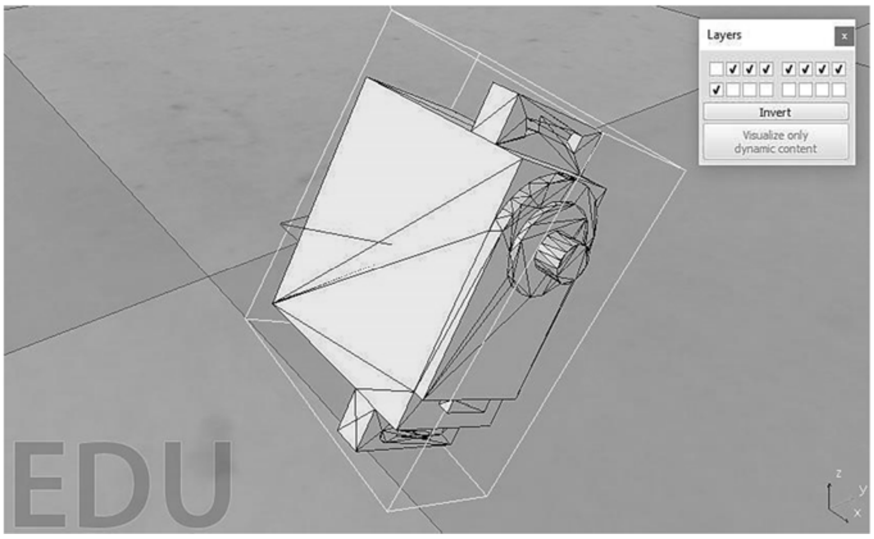


Figure 4.2 – Convex element in V-REP

Convex shape has to be divided by using menu “Edit”, “Grouping/Merging” and “Divide selected shapes”. After that we have several simple shapes, among them the shaft, which we have to use. For applying position of shaft to joint hold down the button "SHIFT" and click on joint and shaft (sequence is important, firstly joint and lastly shaft). After that click “Object/Item Position” icon and in new modal window click “Apply to selection”. The result is shown on figure 4.3.

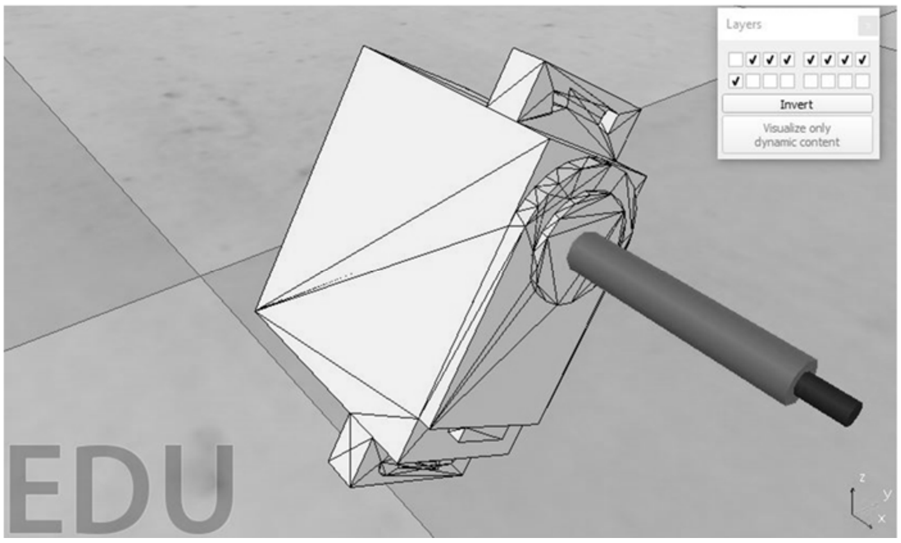


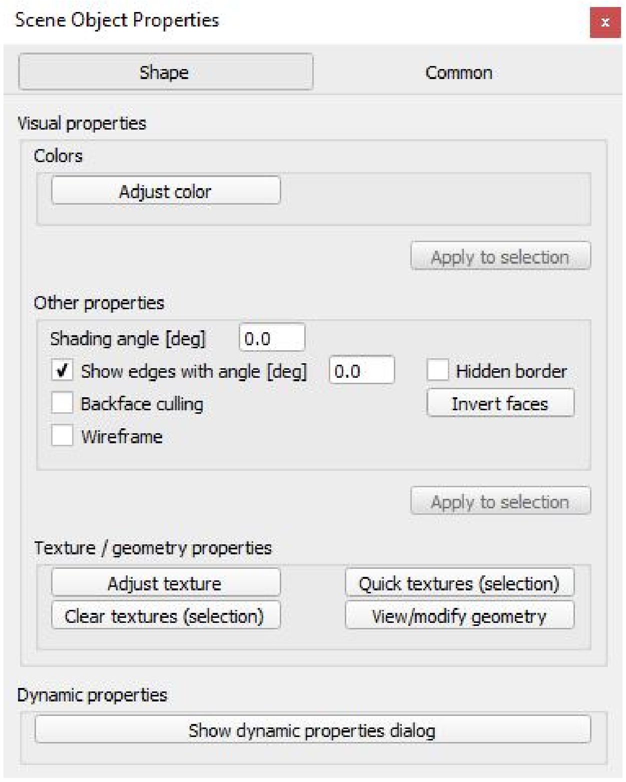
Figure 4.3 – Positioned shaft.

Repeat same sequence of action for all another servos. All joints should be selected and copied to main scenario after correct positioning. In scene hierarchy all joint has to be located appropriately as described above and shown on figure 4.1 (at right). Links in any cases has to be between shapes i.e. can’t be connected to each other.

Different parts of scenario are necessary for simulation of physics and visualization in V-REP. All simple (graphical) shapes have to be duplicated as convex shapes. For creating convex shape based on simple shape, choose the simple shape and use menu “Add”, “Convex decompression of selection”, correct parameters in new modal window and click “OK”. Also convex shapes have to be located in hierarchy appropriately (as shown on figure 4.1 at right). Each convex shape has to be defined as a base for simple shape (which was imported at the beginning). All visible elements were linked with convex shapes (physics) so as the visible shapes is following to physics elements.

4.2 Setting options and properties of shapes and joints

“Scene object properties” is mostly using option in V-REP menu. For each type of item, this menu has two tabs: its own properties (settings) and common setting. For example, convex shape object properties were shown on figure 4.4. Whole menu options are described in V-REP user manual [24].



Shading angle [deg] 0.0
✓ Show edges with angle [deg] 0.0
☐ Hidden border
☐ Backface culling
☐ Wireframe

Figure 4.4 - Convex shape object properties

In the beginning it is necessary to make all convex shapes as a dynamics and “respondable” using menu scene object properties. In Dynamic properties dialog enable checkbox of “Body is respondable” and “Body is dynamic”. Also by clicking onto “Compute mass and inertia properties for selected convex shape” and enter the body density, which for this case equal 1, after that click “OK”. The joints is also has to be corrected. Select joint and open scene object properties and check mode, it has to be "Torque/Force mode". Also here a visual properties "length" and "Diameter" can be changed as it suitable for us.

V-REP contains programming part, without it functionality of software would not be complete. As mentioned above, V-REP use Lua scripts as a default. In V-REP environment there are two types of scripts: threated and non-threated (the difference can be checked in). Non-threaded script was used for this case. For adding have to be used menu "Add", but before select base element in scene hierarchy. Use menu "Add", "Associated child script" and "Non-threaded". All the programming part has to be in this this script for our case. If more exactly, then in script we have four parts and all parts of program have to be written inside of them. The program blocks were shown below with comments.

```
if (sim_call_type==sim_childscriptcall_initialization) then
        -- Some initialization code here
end
if (sim_call_type==sim_childscriptcall_actuation) then
        -- Main ACTUATION code here
end
if (sim_call_type==sim_childscriptcall_sensing) then
        -- Main SENSING code here
end
if (sim_call_type==sim_childscriptcall_cleanup) then
        -- Some restoration code here
end
```

All parts of script with tips for each part were shown above. All joints has to be initialized in first part for our case. Snippet is shown belong. It was shown since can be difficult to understated due to specific determining (in scene hierarchy were used special names to make initialization in loop and use arrays).

```
for i=1,10,1 do
  jointsname[i]='joint_'..i
  jointshandle[i]=simGetObjectHandle(jointsname[i])
  simSetJointTargetPosition(jointshandle[i],jointscurrentpos[i])
end
```

All variables were necessary for manage the joints in scene (for more details check source code to check the appendix A).

4.3 Graphs in V-REP

V-REP allows to use data of simulation and to get information from sensors fixed on movable parts of robot or another object in scenario. Graphs are a convenient tool for data representation.

Graphs are objects that can be used to record, visualize or export data from a simulation. They are very powerful and flexible. The user can select from a multitude of data types applied to specific objects to record. Data is recorded as a data stream (sequential list of data values) that can be visualized in three different ways: time graphs, X/Y graphs, 3D curves. For this case time graph was used.

Graphs can be added in user interface with commands “Add” and “Graph”. To output data on the created graph, it has to be configured appropriately. Select current graph and open “scene object properties”, click “add new data stream to record” and select type of data stream and Object/item to record.

For this case is necessary to add two data streams: firs should be “Force sensor: force along Z axis”, object – appropriately force sensor; second: “Object: absolute Alfa Orientation”, select object “link_1” and press “OK”. After that, click “Edit X/Y graphs” and will be visible a new popup menu as shown on figure 4.6.

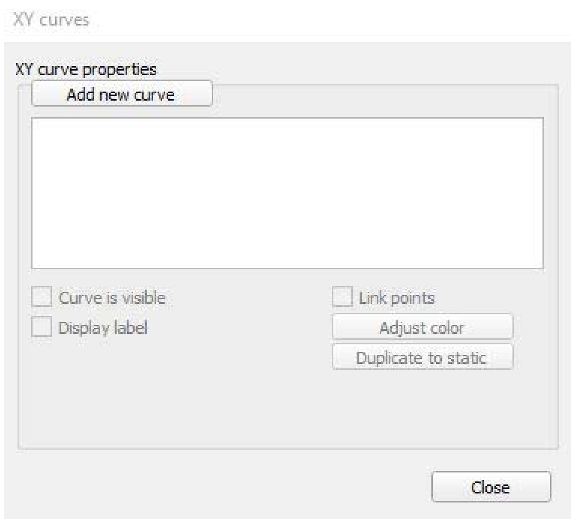
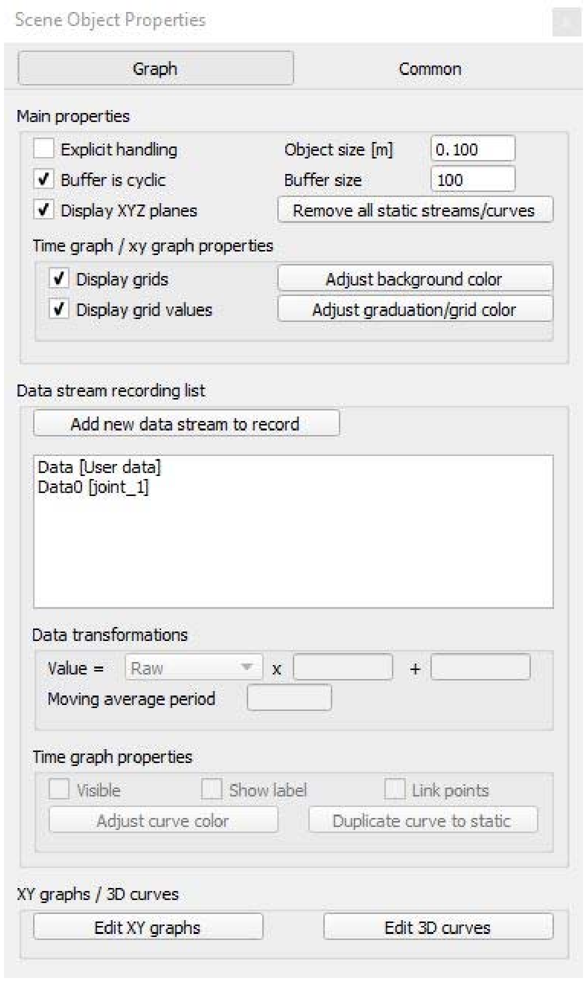


Figure 4.5 - Convex shape object properties

Click “add new curve”, select Data streams and click “OK”, after that add one more cure with appropriately values. When both curves were available, popup menu has to be closed by clicking button “Close”. A result, which has to be after that, was shown on figure 4.7.
The use of graphics is possible in static (when the graph is displayed after completion of the program of simulation), and in the plotting mode during the simulation.


Scene Object Properties
Graph
Common
Remove all static streams/curves
Adjust background color
Adjust graduation/grid color
Add new data stream to record
Adjust curve color
Duplicate curve to static
Edit XY graphs
Edit 3D curves

Main properties

- [ ] Explicit handling
- [✓] Buffer is cyclic
- [✓] Display XYZ planes

Object size [m] 0.100
Buffer size 100

Time graph / xy graph properties

- [✓] Display grids
- [✓] Display grid values

Data stream recording list

Data [User data]
Data0 [joint_1]

Data transformations

Value = Raw x +
Moving average period

Time graph properties

- [ ] Visible
- [ ] Show label
- [ ] Link points

XY graphs / 3D curves

Figure 4.6 – Graph object properties

Graph was configured. For showing the graph has to be associated with “Floating view”. By right clicking on mouse with pointer somewhere of main 3D window, has to be selected “Add” and “Floating view”. After that select graph in scene hierarchy, hover over on floating view, click right button of mouse and in popup menu click “View”, “Associate view with selected graph”. If all procedure were completely finish, then data from source will be outline on graph in floating view.

In addition data stream can be imported from another device or program. For this use next code in main loop:

```
simSetGraphUserData(graphHandle,'Data',difference)
```

Where “graphHandle” is handle of graph, “Data” is name of stream and “difference” is difference between previous and next value.

4.4 Sensors in V-REP

In V-REP available three types of sensors: Proximity sensors, Vision sensors and Force sensors. The most commonly used and popular is a force sensor. Force sensors are initially rigid links between two shapes that are able to measure transmitted forces and torques. The rigidity of force sensors is conditional, in the sense that force sensors can be broken if a certain condition arises (e.g. if a force or torque threshold was exceeded).

As mentioned, force sensors are initially rigid links between two shapes, that means that force sensor has to be between two dynamically shapes anyway. Force sensor can be added using menu “Add”, “Force sensor”. Same command for another type of sensors. In our case only position sensors was used for getting of servos position (revolute connections). For this appropriate “joint handles” has to be added in non-threated script and using next command we can get data about current positions.

```
out=simGetJointPosition(jointsHandle)
```

The unit of getting data is radians that should be taken into account every time.

4.5 Simulation features

Real characteristics of servo are not too simple as we like to get and anyway significant simplification will exist.

Simulation process the behavior of servo requires a lot of attention, but no reasons to expect particularly good results from this is not required. V-REP allows using PID controller (proportional integral derivative controller) in simulation and with appropriate parameters may be quite effective. But for our case it is not possible since we are using low-cost servos with very cheap electronic components.

There is another approach for simulation of servo behavior. This approach based on the characteristic dependence of the rate of rotation of the shaft of the servo from time to time (shown on figure 4.7). This characteristics was taken as shown on [24] by quite popular manufacturer of servo motors in USA. From next formulas we can determine all requirements for approximately simulation of servo behavior.

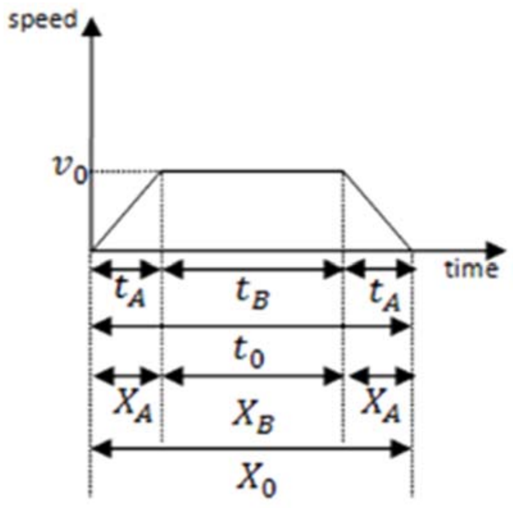


Figure 4.7 – Trapezoid Operating Pattern

$v_0 = \frac{X_0}{t_0 - t_A}$ – maximum speed, $t_A = \frac{t_0 - X_0}{v_0}$ – acceleration/deceleration time, $t_B = t_0 - 2 \cdot t_A$ – constant-velocity travel time, $X_0 = v_0(t_0 - t_A)$ - total travel distance, $X_A = \frac{v_0 \cdot t_A}{2}$ – acceleration/deceleration travel distance, $X_B = v_0 - t_B$ – constant-velocity travel time.

Special algorithm was developed in V-REP for simulation of servo movements considering the above described specifics of servo (block diagram was shown on figure 4.8).

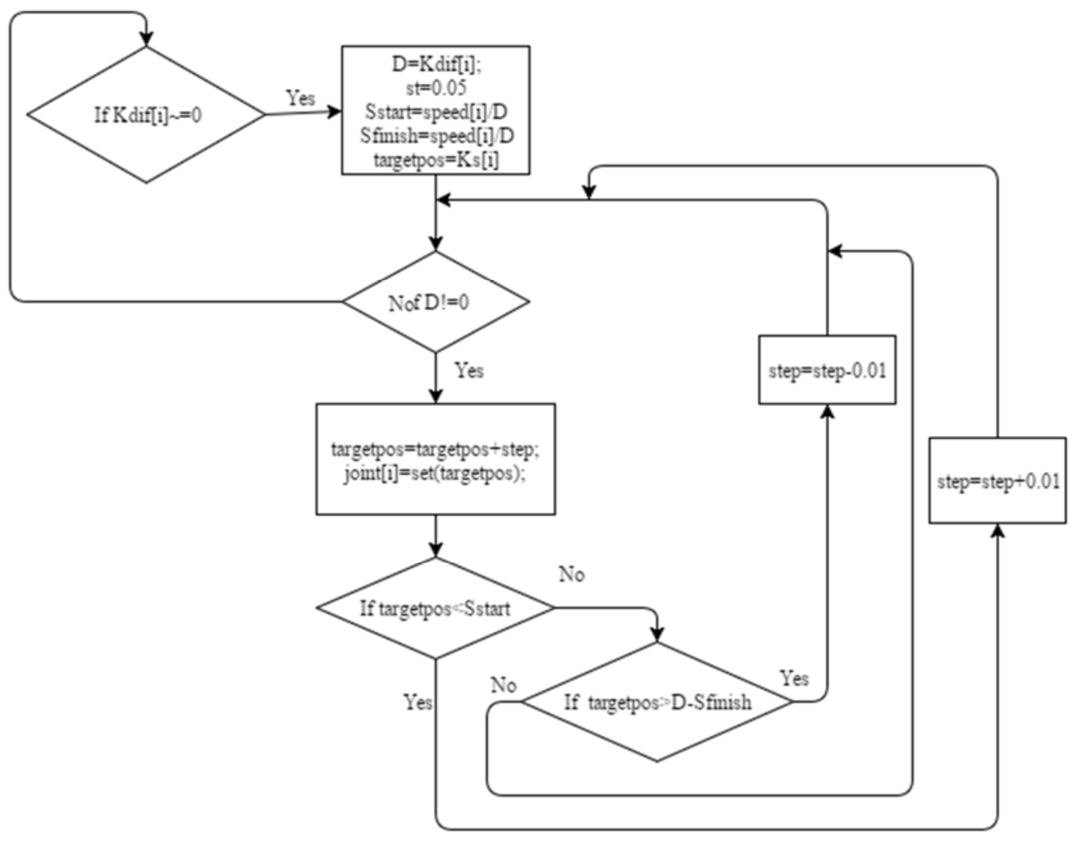


Figure 4.8 – Block diagram of algorithm for servo simulation

As shown on block diagram, the acceleration and deceleration time depends on maximum speed and time of movement also depends to maximum speed. This algorithm does not take into account the moments of forces acting on the shaft of the servo. This algorithm was developed considering that the servos are running at low speeds.

5. CREATING INTERFACE(S) BETWEEN A PC AND THE ROBOT

Interfacing between PC and developed robot is delivered using Virtual Robot Experimentation Platform (V-REP) through the serial connection, i.e. connection to Universal Serial Bus (USB). Special algorithm was created for providing control of robot and simulation of robot movement in same time. For transmit data from the microcontroller to computer were developed codes in both side. In V-REP code was written in the functional part that connects to the port and reads the data. On microcontroller also the algorithm was appropriately realized to communicate by serial port.

5.1 Data communication

As mentioned, data communication was realized by serial communication, which is quite slow for data transmitting but which is simple and commonly used method. Probably students already had experience of using this type of communication and there are not needed to explain the principles of operation. Serial communication has own features which has to be taken into account during the whole time of using. First of them is unavailability controlling continuity of data. Some data can be missed and it is not controlled by interface of data transmission.

For correct data exchange the algorithm of data transfer was developed (block diagram was shown on figure 5.1), which provides transfer of data in a strict sequence. So as all values has to be sent at same time, has to be used some separator for arrays of data. The comma was used as a separator. Next fragment of program shows the organization of setting commands from microcontroller.

```
Serial.print(r3); // add value at end of string
Serial.print(",");//data separator
Serial.print(l3); // add value at end of string
Serial.print(",");//data separator
```

As shown above, for all parameters except the last one. When writing the last parameter, use the write command with a newline. When executing the command "end of line" in the series connection is sent special character symbol that we use as the token-separator between the data streams.

Serial.println(l2); // add value at end of string and make new line (i.e. transmit end of line character).

Quite difficult processing algorithm has to be realized on V-REP side unlike microcontroller side. Next processing algorithm provides all that needed for getting data correctly on V-REP side.

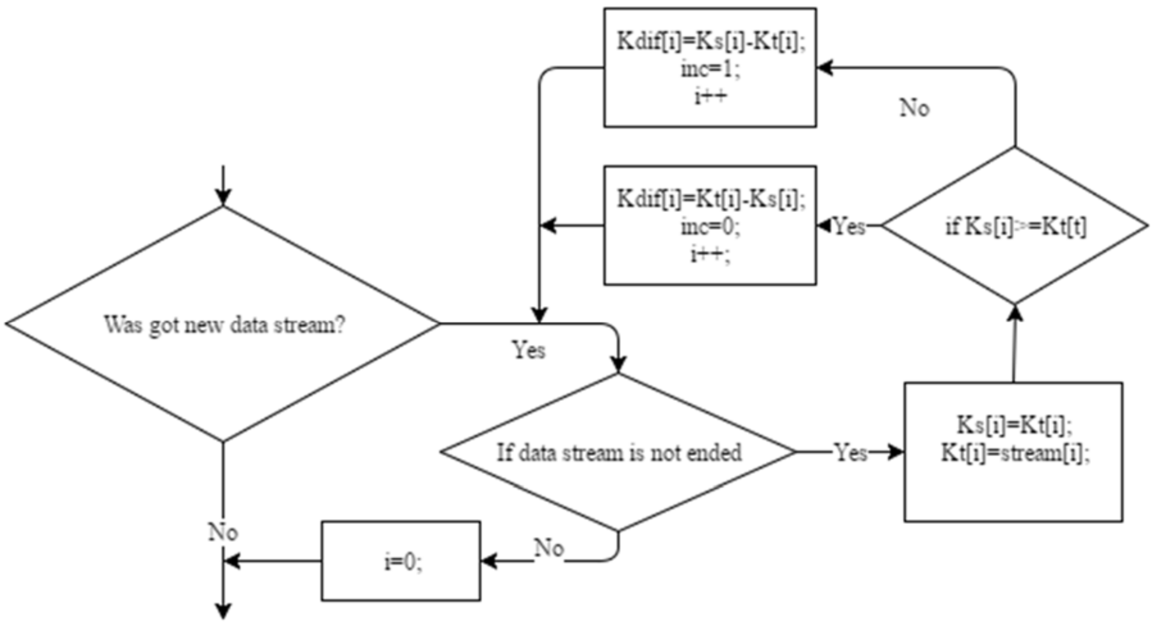


Figure 5.1 –Block diagram of data reading algorithm

5.2 Servo controller and control scheme

Arduino Uno was used as a controller. Arduino is most popular device for controlling of servos and familiar even to beginners in robotics. Also Arduino is quite simple device and don't require many times for studying, for cases when students don't have skills.

Special Arduino library [25] for servos was used in project for controlling of servos with different speed. The "VarSpeedServo.h" Arduino library allows the use of up to 8 servos moving asynchronously (because it uses interrupts). In addition, you can set the speed of a move, optionally wait (block) until the servo move is complete, and create sequences of moves that run asynchronously.

Empty board of Arduino UNO was used as a base for creation of servo connection circuit.

It is very easy and available method, because it provides a reliable connection to Arduino all the ports with minimal effort and low cost. To create the connection diagrams on the basis of such fees requires minimal skills with a soldering iron and a little time.

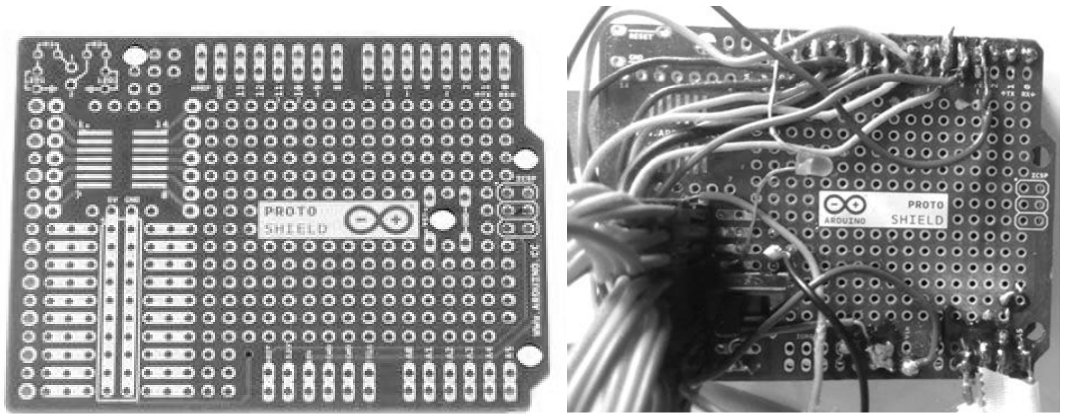


Figure 5.3 – Empty board of Arduino UNO (left) and ready board for biped robot which was used (right).

This type of work can be done by students as a one part of work for getting experience in modifying and creation of circuits.

PSB (printed circuit Board) was developed for cases, when need a many circuits. Also, this work could be done by students since it is not quite big deal and all needed information about it and tools are available, but it takes much more times.

On figure 5.4 was shown scheme of PCB which was prepared in Sprint Layout software. This software has quite friendly user interface.

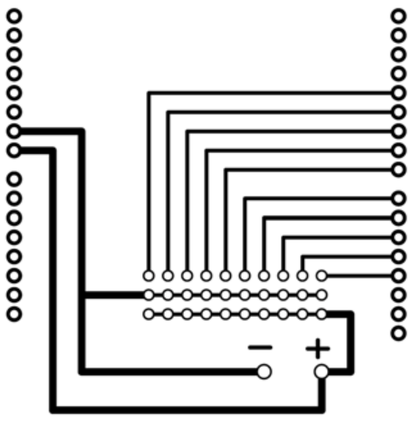


Figure 5.4 – PSB can be used instead empty Arduino board

5.3 Modifying of electronic components (sensors)

The lack of control at the ground surface contact makes a walking robot under actuated. Therefore, a bipedal robot may fall due to an external disturbance, or due to an incorrect action of the robot itself. For described reason above, the biped robot has to have four pressure sensors on each contact surface, it allows make balancing dynamically. But also the biped robot can operate without them for some cases and work without dynamically balancing. Signals from these sensors allow preventing biped robot from falling. In this case were not used the sensors on biped robot, but the mentioned sensors were simulated.

For each servo was realized feedback, which allow to get real servo position for every time. Feedback was realized using standard potentiometers of each servos, the signal was processed after setting position with high precision.

Servos were disassembled in order to implement the feedback. Using multimeter was determined which of the three contacts is the servo position signal. After that gray wire was connected to this contact (on figure 5.3), in the hull of servo was made a hole through which the wire was brought out.

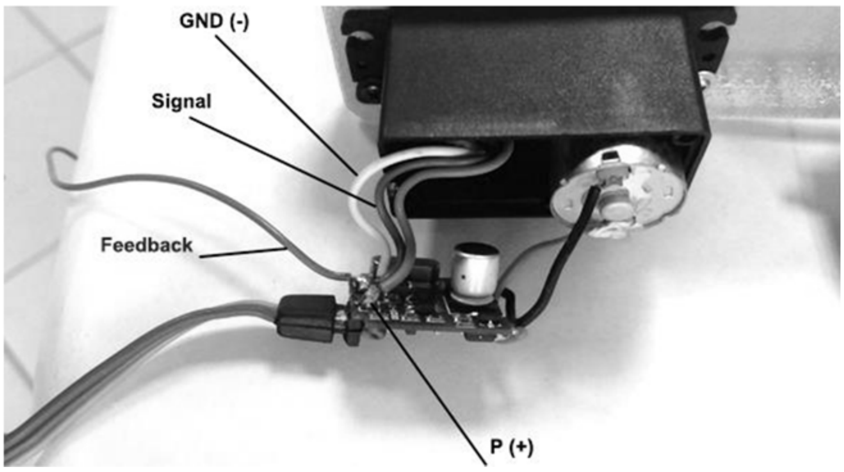


Figure 5.3 – Feedback from servo

On this wire we are getting the analog signal (voltage), which value is changing depending to the position of servo from some minimum value to maximum value. For each servo this minimum and maximum was different and it should be taken into account in the future. After all the operations we have feedback and can use it after assembly of the hull of servo.

6. TESTING (COMPARISON OF THE VIRTUAL AND THE PHYSICAL ROBOT)

Testing results are one of the most important steps which aim to evaluate the results and determine errors. Compare the virtual and physical robot was included in the aims as a most interesting and important indicator. Also were analyzed changes of parameters where is not possible to conduct a comparative analysis because this work had a limitations.

6.1 «Pre targeting» mode

As mentioned above the «pre targeting» mode means that the controller sending target positions to real servos and virtual servos at same time. The advantage of this mode is the ability to testing of algorithms written without using the real robot. This is a very important aspect when writing the control algorithms for the inexperienced students who can make critical mistakes and to break the robot often. Using of 3D printable parts making the process quite cheap but it is not too fast and should be one mode, when we can make pre testing and find main critical mistakes.

On figure 6.1 was shown simplified block illustration of communication for pre targeting mode.

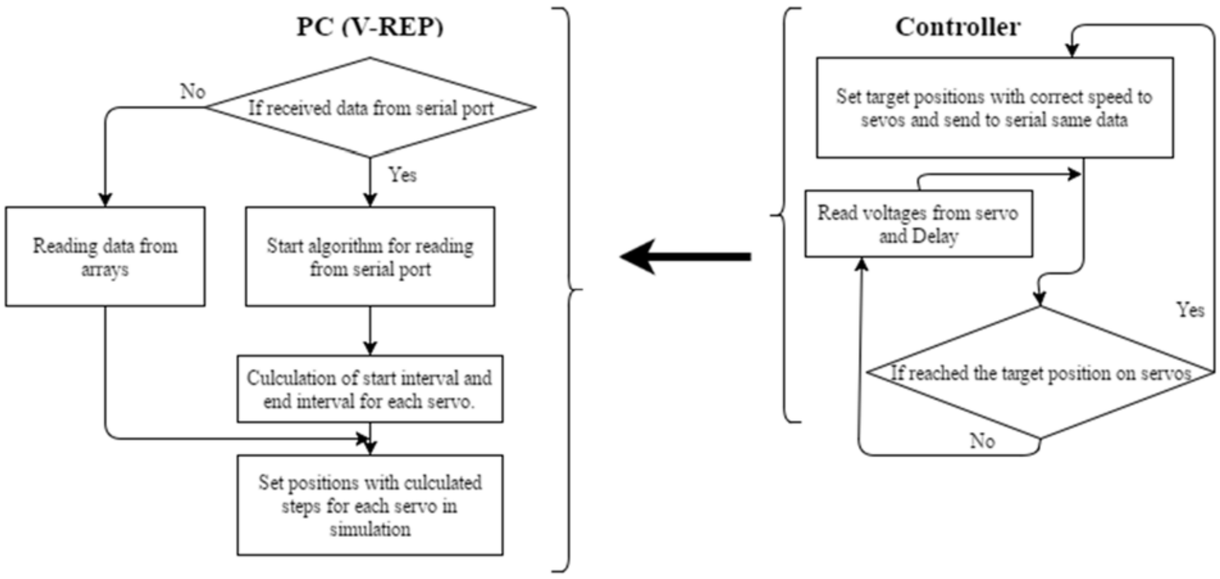

PC (V-REP)
If received data from serial port
No
Yes
Reading data from arrays
Start algorithm for reading from serial port
Culculation of start interval and end interval for each servo.
Set positions with culculated steps for each servo in simulation
Controller
Set target positions with correct speed to sevos and send to serial same data
Read voltages from servo and Delay
If reached the target position on servos
Yes
No

Figure 6.1 – «Pre targeting» algorithm illustration of communication

On Controller side was realized setting of positions (and speeds also) for real servos and sending of same data to serial communication at same time. The approach described above doesn't reflect the real situation with physical robot, but allow testing of controller algorithms.

Testing for this mode is very challenging due to the complexity of the mapping data simulations and data from physical model, therefore testing was conducted 10 times. For testing was written a simple algorithm sitting down and getting up the robot. For testing was written a simple algorithm sitting down and getting up the robot. When performing the specified algorithm, at the same time includes 4 servo mode out-of-sync. Data were obtained in the V-REP software from simulation and using the built-in servo potentiometers (as described above, we connected the wires to them and connected to the microcontroller).

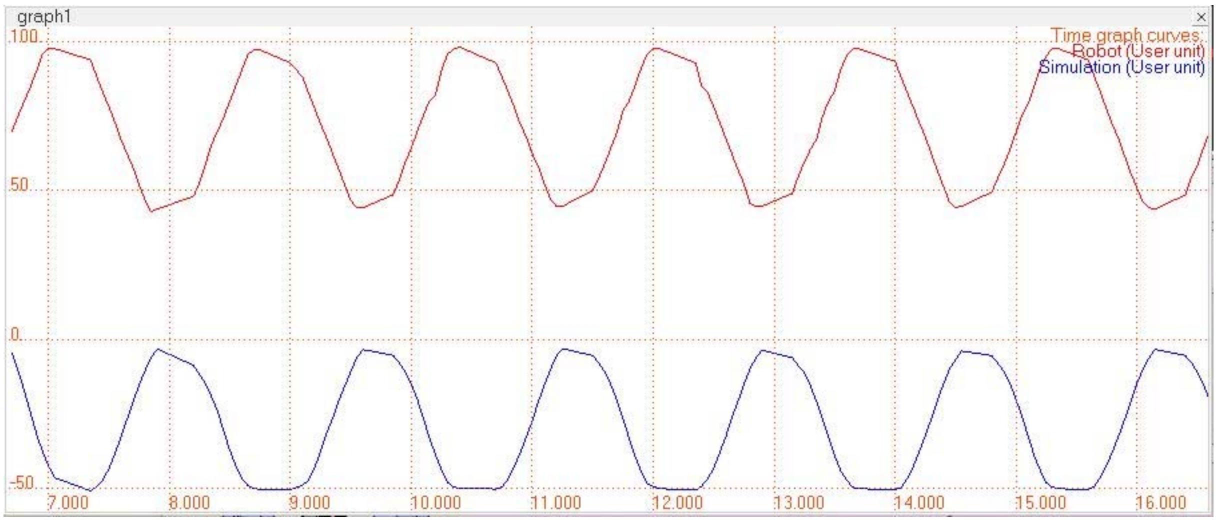


Figure 6.2 – Graph of real servo positions and simulated servo positions was made in V-REP.

Also on figure 6.2 was shown how changing the angles value for Servo1. The synchronization of the simulation and the real servo has not been verified during testing so as it doesn't meter in this mode (reasons mentioned above). On figure 6.2 the changing of values for the real servo was shifted in phase by $\frac{\pi}{2}$ because the data was sent from the controller with a delay. The crux in the fact that having one serial connection, we cannot send the data at different times, so the data taken during the rotation of servo and this data were sent together with the parameters of servo for next iteration.

The volts were translated to degrees by the formula:

$$P_{cur} = (x - V_{\min}) \cdot K + P_{min}$$

P_{min} – value of angle at the minimum position, V_{min} – value of voltages at same position P_{min}, x- current value of voltages, K - to change the voltage on 1 unit, it is necessary to change the position on this (K) angle, K was calculated as $K = \frac{Vmax-Vmin}{Pmax-Pmin}$; The reason for this approach was that the V_{min} and V_{max} can be different for each case, but there can increase or decrease on same value.

The formula of translate radians to degrees (since in V-REP simulation has angle values in radians) was used for obtaining of values shown in figure 6.2.

$$a = \frac{x \cdot 180}{\pi}$$

Where x – value of angle in radians, π - mathematical constant, the ratio of a circle's circumference to its diameter.

As we can see in figure 6.2 the servos taking some times between commands (between standing up and sitting down). The reason for this is a necessity of some time for sending data to serial connection. This problem is not important for our case, since this factor was considered in simulation algorithm.

On figure 6.2.2 was shown recalculated data for simulation. Also this curve was shifted little bit down for much more visibility. Special macros by VBA (Visual Basic Application) in EXCEL [26]software also was created for using exported data from V-REP.

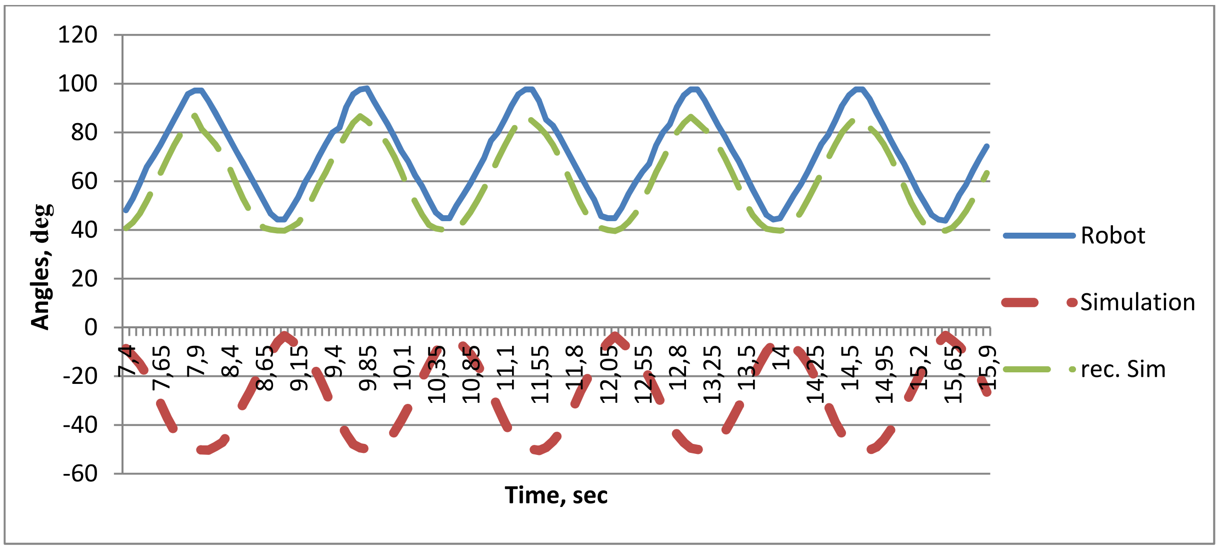


Figure 6.2.2 – Comparatively graph of robot and simulation without shifting was made in EXCEL software.

6.2 «Real targeting» mode

«Real targeting» mode is simpler mode than pre targeting, which has next main idea – simulation in real time. It means that the simulation is supporting only on data about current position of servos. For this mode was the scenario of simulation same to the pre targeting mode, but algorithm was developed different to the mentioned mode.

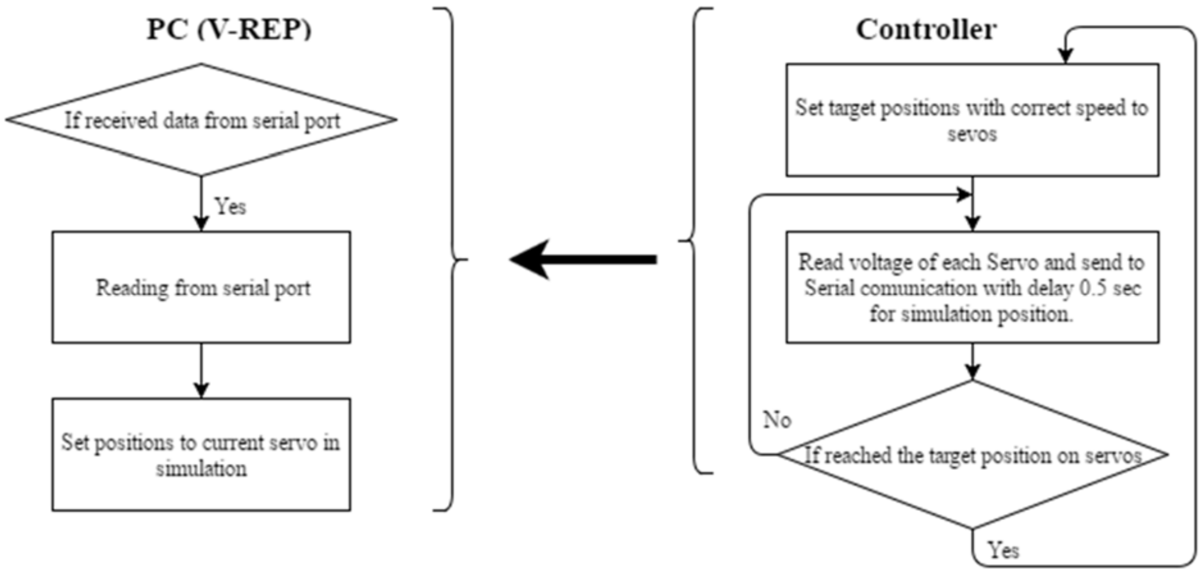

PC (V-REP)
If received data from serial port
Yes
Reading from serial port
Set positions to current servo in simulation
Controller
Set target positions with correct speed to sevos
Read voltage of each Servo and send to Serial comunication with delay 0.5 sec for simulation position.
No
If reached the target position on servos
Yes

Figure 6.3 – Real time algorithm illustration of communication between simulation and robot

On figure 6.3 was shown illustration of communication algorithm between V-REP and controller. As can be detected the simulation will be static for cases when voltages is constant. The formulas were used same as in the previous algorithm to convert data into order to display data on graph. Formulas also were used for translation of voltages to angles. Also was used special translating operation of voltages for two servos due to there was located differently and were inverted to each other (check in appendix B).

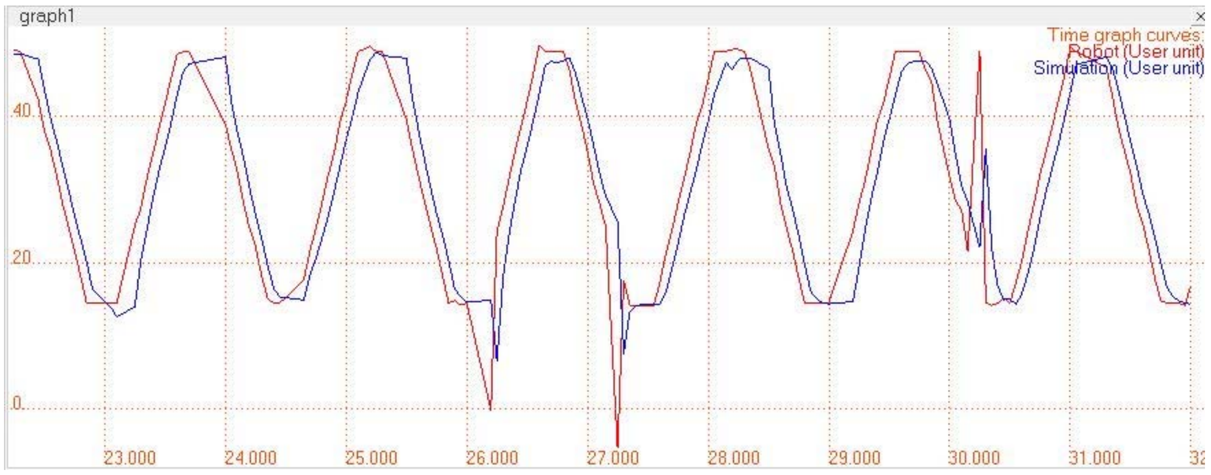


Figure 6.4 – Real time algorithm illustration of communication between simulation and robot

On figure 6.4 was shown the graph of data from simulation and form real robot. There is time difference between simulation and real robot which can be determined from graph 6.4 but it has small value.

Figure 6.4 shows the value of angles in degree units from one joint of simulation and same joint from real robot. As we can see, features of this graph are fluctuating signal we get from the servo. The value of them was quite high and has to be taken into account, but in simulation this fluctuations were lowed. The fluctuation leads to crash of simulation sometimes even it was lowed. Therefor to determine the reasons of this fluctuation was determine next possible hypotheses. The errors may be in: V-REP calculations, data communication, controller calculations (Arduino code) or inside servo. Fluctuations are of random nature as can be detected from graph.

For beginning the data of plotting was imported from V-REP and checked. Missing of some data was revealed – this is another problem. Spatial condition was added in algorithm for cases, when data were missed. In cases when were skipped more than 4 data sets successively then simulation fails. A message is displayed. This happens very rarely, but such a condition is necessary.

After that, was made V-REP code for outputting data as a voltage value without calculation and transformation. On figure 6.5 was shown the results. Problem is not in recalculation of voltages to angles.

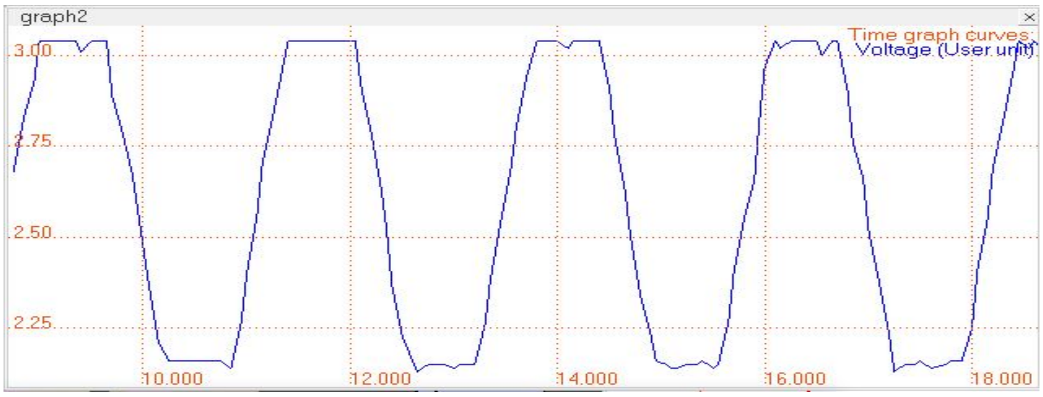
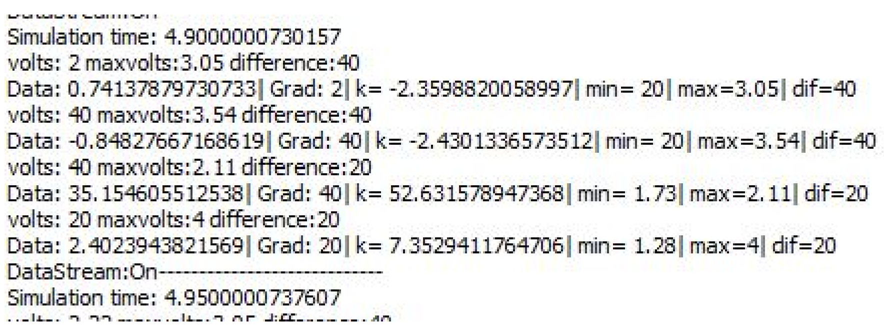


Figure 6.5 – Voltage data in V-REP after getting from serial connection

Outputting of data after reading was next step. Interruption of data stream was revealed for some cases and continuing in next iteration. This interruption are mixing variables and getting not correct data. Result from debugging mode was shown on figure 6.6.



```
Simulation time: 4.9000000730157
volts: 2 maxvolts:3.05 difference:40
Data: 0.74137879730733| Grad: 2| k= -2.3598820058997| min= 20| max=3.05| dif=40
volts: 40 maxvolts:3.54 difference:40
Data: -0.84827667168619| Grad: 40| k= -2.4301336573512| min= 20| max=3.54| dif=40
volts: 40 maxvolts:2.11 difference:20
Data: 35.154605512538| Grad: 40| k= 52.631578947368| min= 1.73| max=2.11| dif=20
volts: 20 maxvolts:4 difference:20
Data: 2.4023943821569| Grad: 20| k= 7.3529411764706| min= 1.28| max=4| dif=20
DataStream:On---------------------------
Simulation time: 4.9500000737607
```

Figure 6.6 – Result from debugging mode (V-REP)

So as we are reading data through the serial connection we know how much variables we have to get. For this reason condition was added in algorithm of reading. This condition not allows applying read data for using variables for cases when data stream has some interruption and in fact interrupted data are miss. It is allow since we are using quite small step of simulation. On figure 6.7 was shown result from V-REP after correction of algorithm.

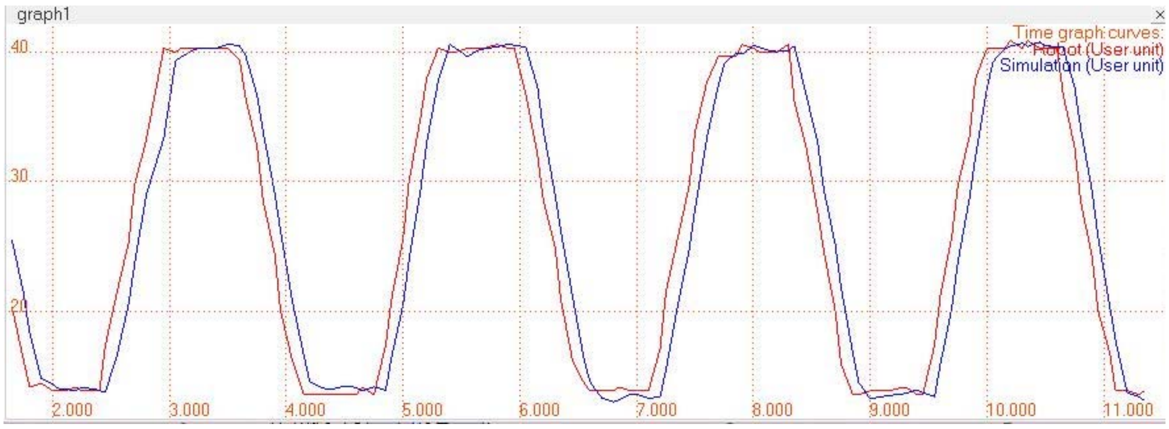


Figure 6.7 – Results of finally algorithm in V-REP

7. CONCLUSION

Features of bipedal robots development were determined. Simplified biped robot was developed based on determined features. This robot is quite simple and saves main principles of biped robots in order to be used as a teaching tool in robotics. The robot used low-cost R/C servos and 3D printable parts instead of industry-grade components. This platform is cheapened to the maximum and emphasizes the imperfections, which must be modeled and analyzed by students in studying process. This platform was developed inaccurately since it is not required for the assigned tasks.

Available options for simulation of robotic systems were revealed and simulation tool for studying platform was determined. Two main algorithms of simulation were developed. Each mode has application tasks. “Pre targeting” mode is quite useful for algorithms’ testing on first stage of biped robot development. “Real time” mode is helpful in getting current information from real robot and analysis.

Testing was conducted to evaluate the accuracy of the simulation. Testing showed that the results of simulation is close to the present robot’ output data in more than 94% situations for real time simulation mode and more than 96% for pre targeting mode. It is quite good result for teaching tool. The simulation can be improved by entering adjustments if necessary since the obtained errors are mostly constant. Any improvement could bring at most 2% in accuracy and require significant labor effort that is why optimization was not conducted.

7.1 Collisions and problems

During the development of biped robot, creation of simulation, testing and other steps different problems were detected.

One of them was the problem with V-REP. The problem was that there was fail to read data through serial connection while this connection gave the positive result (i.e. connection was available, but worked incorrect).

This problem was solved by editing configuration file in V-REP directory. File usrset.txt located in folder named as “system” in “V-REP” directory. Changing parameter value of “useAlternateSerialPortRoutines” to “On” in usrset.txt solved this problem.

This problem is related to operating system of computer (in this case it was “Windows 7-8-10).

Hardware – problems with position precision of servo. The used servos were not absolutely the same and for servos it was necessary to set different values of angle for getting the same change of rotation (the same positions). It is highly recommended to take a servo of the same model and same production company.

Arduino Uno has not enough analog ports, also it is recommended taking Arduino Mega for next times. Adruino Mega has several serial connections unlike Arduino Uno which has only one serial connection. Arduino mega costs a little bit more, but can solve important problems with complexity of programming at simultaneous transmission of data for pre targeting mode.

7.2 Future work

This work had many limitations due to complexity of biped robot development. Also there was limitation of economic factor so as it is very important factor in real life. In the future it is planned to conduct research using of force and vision sensors and do tests to evaluate possibility of it’s use.

Also testing of another types (much faster) of communication is quite important and could bring interesting results.

KOKKUVÕTE

Kahejalgsed robotid on väga keerukad mehhatroonilised lahendused, milline asjaolu muudab nende uurimise küllaltki keerukaks. Omandatud teadmiste rakendamine eeldab praktilist harjutamist nii ehtsate robotite kui simulaatoritega. Sedalaadi ülesannete täitmiseks leidub arvukalt sobivaid lahendusi, kuid kõigil on oma varjuküljed. Põhiliseks miinuseks on nende kõrge hind. Käesoleva uurimistöö eesmärgiks on töötada välja kahejalgne pildiedastusvõimalusega robot, mida saaks kasutada robootika õppevahendina.

Kahejalgse roboti väljatöötamise iseloom oli kindlaks määratud. Lihtsustatud kahejalgse roboti väljatöötamisel põhineti selgelt määratletud võimalustel.

Loodud platvorm valmis võimalikult väheste kuludega ja on pisut vigane, mis muudabki selle platvormi õpilastele sobivaks vahendiks, mida õppe käigus analüüsida ja täiustada. Kahejalgse roboti kujundamine toimus ebatäpselt, kuna see ei ole ette nähtud määratud ülesannete täitmiseks.

Toodi välja robootikasüsteemide simuleerimiseks leiduvad valikuvõimalused ja määrati kindlaks õppeplatvormi jaoks kõige täpsem simuleerimisvahend. Simulatsioonisüsteemi tingimuseks oli selle tasuta kasutamise võimalus, kuna majanduslikud tegurid olid väga suure tähtsusega. Me otsustasime tarkvara kasuks, mille nimetus on "Virtual Robotics Experimentation Platform" (V-REP) (Virtuaalne robootika katsetustegevuse platvorm).

Simulatsiooni kahe põhialgoritmi väljatöötamiseks kasutati eelpoolmainitud tarkvara. Esimest algoritmi nimetati „eesmärgieelseks" olekuks (pre targeting mode) ja teist „reaalajas" olekuks (real time mode). Mõlemal olekul on oma kindel rakendusülesanne. „Eesmärgieelne" olek sobib kasutamiseks algoritmide testimisel kahejalgse roboti väljatöötamise esimeses etapis. „Reaalajas" olek aitab saada jooksvat teavet ehtsalt robotilt ja seda analüüsida.

Simulatsiooni täpsuse hindamiseks viidi läbi testimine. Testid näitasid, et stimulatsiooni tulemused on väga sarnased olemasoleva roboti väljundandmetega rohkem kui 94% kordadest reaalajas oleku puhul ja rohkem kui 96% kordadest eesmärgieelse oleku puhul. See on õppevahendi kohta küllaltki hea tulemus. Simulatsiooni saab parandada uute regulatsioonide sisestamise abil, kuna esinenud vead on enamasti muutumatud. Iga täiendus võib lisada täpsust vähemalt 2% võrra ning see nõuaks ühtlasi märkimisväärset lisatööd, mistõttu tulemuste täielik optimeerimine jäi saavutamata. Käesoleva uurimistöö käigus saadud tulemusi võib kasutada mitte ainult ülikoolides, vaid ka robootikakallakuga kesk- ja kutsekoolides, kuna tegemist näib olevat odava õppevahendiga.

REFERENCES

1. *Planning Walking Patterns for a Biped Robot.* **Qiang Huang, Kazuhito Yokoi, Shuuji Kajita, Kenji Kaneko.** 3, s.l. : IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION, 2001, Vol. 17.

2. *Efficient Bipedal Robots Based on Passive-Dynamic Walkers.* **Steve Collins, Andy Ruina, Russ Tedrake, Martijn Wisse.** 1082-1085, s.l. : Science, 2005, Vol. 307, pp. 1082-1085.

3. *Modeling and Control for a Biped Robot on Uneven Surfaces.* **Chen, Jian Li and Weidong.** Shanghai : Joint 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference, 2009.

4. *Poppy project.* [Online] https://www.poppy-project.org/. [Cited: 04 09, 2016.] https://www.poppy-project.org/.

5. **Denamganaï, Kevin.** HaRo: 10 DoF Bipedal Gait. *Instructables.* [Online] [Cited: 02 24, 2016.] http://www.instructables.com/id/HaRo-10-DoF-Bipedal-Gait/.

6. **Böttcher, Sven.** Principles of robot locomotion. *Seminar 'Human robot interaction'.*

7. **S.Roland.** *Introduction to autonomous mobile robots.* 2004. pp. 12-45.

8. **Dekker, M.H.P.** *Zero-moment point method for stable biped walking.* Eindhoven : s.n., 2009.

9. *Zero Moment Point – Thirty five years of its life.* **M. Vukobratovic, B. Borovac.** s.l. : International Journal of Humanoid Robotics, pp. 157–173, 2004, Vol. vol. 1.

10. *Zero Moment Point/Inverted Pendulum-Based Walking Algorithm for the NAO Robot.* **Iulia M. Motoc, Konstantinos Sirlantzis, Sarah Spurgeon, Peter Lee.** [ed.] Fifth International Conference on Emerging Security Technologies. Washington : IEEE Computer Society Washington, DC, USA ©2014 , 2014. pp. 63-66. ISBN 978-1-4799-7007-0.

11. **E. Cuevas, D. Zaldivar, R. Rojas.** *Walking trajectory control of a biped robot, Technical report.* Berlin : Freie Universität, 2004.

12. *Postural Stability of Biped Robots and the Foot-Rotation Indicator (FRI) Point.* **Goswami, Ambarish.** Pennsylvania : Department of Computer and Information Science, University of Pennsylvania, Philadelphia, 1999. 19104-6389.

13. ASIMO Tehnical Information. *ASIMO Honda.* [Online] Honda Motor Co., Ltd., 2007. [Cited: 03 19, 2016.] http://asimo.honda.com/downloads/pdf/asimo-technical-information.pdf.

14. *Contact impact inhibition strategy for biped robot walking based on.* **Zhipeng Wang, Bin He, Runjie Shen and Weibin Meng.** Zhuha : IEEE Conference Publications, 2015. 978-1-4673-9675-2/15/.

15. *Kinematic Analysis and Motion Planning of a Biped Robot with 7-DOF and Double Spherical Hip Joint.* **Guangri Li, Qiang Huang, Yanping Tang, Guodong Li and Min Li.** Beijing : Proceedings of the 7th World Congress on Intelligent Control and Automation, 2008.

16. **Engineering department, Cambridge University.** Mechanics Formulae and Data. [Online] 2000. [Cited: 01 13, 2016.] http://www-mdp.eng.cam.ac.uk/web/library/enginfo/cueddatabooks/mechanics.pdf.

17. **Luksch, Tobias.** *Human-like Control of Dynamically Walking Bipedal Robots.* Kaiserslautern : Technischen Universit at Kaiserslautern, Technischen Universit¨ at Kaiserslautern, 2010.

18. **Margaret A. McDowell, Ph.D., M.P.H., R.D., et al., et al.** *Anthropometric Reference Data for Children and Adults: United States.* s.l. : National Health Statistics Reports, 2003–2006. Number 10, October 22, 2008.

19. **Karl Muecke, Jeff Kanetzky, Raghav Sampath and Patrick Cox.** Design of a Bipedal Walking. *Stevens Institute of Technology.* [Online] 2006. [Cited: 11 14, 2015.] https://web.stevens.edu/msrobotics/SMRDC2010/muecke06r.pdf.

20. Servo Database . *Servo Database - TowerPro Servos.* [Online] [Cited: 01 29, 2016.] http://www.servodatabase.com/servos/towerpro.

21. MG995 High Speed Metal Gear Dual Ball Bearing Servo. *Electronicos Caldas.* [Online] [Cited: 02 09, 2016.] http://www.electronicoscaldas.com/datasheet/MG995_Tower-Pro.pdf.

22. *Coppelia Robotics V-REP.* [Online] Coppelia Robotics. [Cited: 11 04, 2015.] http://www.coppeliarobotics.com/.

23. *Gazebo.* [Online] Open Source Robotics Foundation. [Cited: 01 06, 2016.] http://gazebosim.org/.

24. Servo Motors and Drives. *Anaheim Automation.* [Online] [Cited: 02 19, 2016.] http://www.anaheimautomation.com/manuals/forms/servo-motor-guide.php.

25. VarSpeedServo. *GitHub.* [Online] [Cited: 01 23, 2016.] https://github.com/netlabtoolkit/VarSpeedServo.

26. Spreadsheet Software Programs - Excel. *Microsoft Office.* [Online] [Cited: 03 09, 2016.] https://products.office.com/en-us/excel.

APPENDIX A

Source code in V-REP

```
if (sim_call_type==sim_childscriptcall_initialization) then
    base=simGetObjectHandle('robot_dyn')
    graphHandle=simGetObjectHandle("graph1")
    jointsname={}
    jointshandle={}
    jointscurrentpos={}
    jointstargetposition={}
    jointsintervalspeed={} -- contents the Sstart and Sphinish
    jointsdifference={}--difference between target and current
    jointsaction={}--0-decrease,1-increase
    statjointsdifference={}
    steps={}
    k=1
    for i=1,10,1 do
        jointsname[i]='joint_'..i
        jointshandle[i]=simGetObjectHandle(jointsname[i])
        jointscurrentpos[i]=0
        jointstargetposition[i]=0
        jointsdifference[i]=0
        jointsaction[i]=0
        jointsintervalspeed[i]=0
        statjointsdifference[i]=0

simSetJointTargetPosition(jointshandle[i],jointscurrentpos[i])
        steps[i]=0.01
    end
    --- include the COM port and start of SERIAL connection
     portNumber="\\\\.\\COM3"
    --could be defined as followed
    --portNumber=[[\\.\COM12]]
    baudrate=38400
    serial=simSerialOpen(portNumber,baudrate)
    inpos=0
    voltin=1
    cnt=0
    val={}--contains data from data stream
    volts={}--contains data from data stream
    for i=1,30,1 do
        val[i]=0
    end
    for i=1,13,1 do
        volts[i]=0
    end
end
if (sim_call_type==sim_childscriptcall_actuation) then
    --get information from serail connection and separate of
```

```
them---
    str=simSerialRead(serial,200,false,'\n',200)
    if str ~= nil then
        local token
        cpt=0
        --extracting the values in str separated by a ,
        for token in string.gmatch(str, "[^,]+") do
            cpt=cpt+1
            val[cpt]=token
        end
        for i=1,10,1 do

jointstargetposition[i]=(tonumber(val[i]))*math.pi/180
            if jointstargetposition[i]>jointscurrentpos[i]
then
                jointsaction[i]=1
                jointsdifference[i]=jointstargetposition[i]-
jointscurrentpos[i]

statjointsdifference[i]=jointstargetposition[i]-
jointscurrentpos[i]
            end
            if jointstargetposition[i]<jointscurrentpos[i]
then
                jointsaction[i]=0
                jointsdifference[i]=jointscurrentpos[i]-
jointstargetposition[i]
                statjointsdifference[i]=jointscurrentpos[i]-
jointstargetposition[i]
            end
        end
        for i=11,20,1 do
            jointsintervalspeed[i-10]=tonumber(val[i])/20
        end
        for i=21,32,1 do
            volts[i-20]=tonumber(val[i])
        end
        voltin=1
        for i=1,10,1 do
            steps[i]=0.02
        end
    end
    for i=1,10,1 do
        while true do
            if (jointsdifference[i]<0 or
jointsdifference[i]==0) then
                break
            end
            if (statjointsdifference[i]-
jointsdifference[i]<jointsintervalspeed[i]) then
                steps[i]=steps[i]+0.02
            end
            if (jointsdifference[i]<jointsintervalspeed[i])
then
                steps[i]=steps[i]-0.02
            end
            if (jointsaction[i]==0) then
                 jointscurrentpos[i]=jointscurrentpos[i]-
steps[i]
            end
            if (jointsaction[i]>0) then

jointscurrentpos[i]=jointscurrentpos[i]+steps[i]
            end
            jointsdifference[i]=jointsdifference[i]-steps[i]

simSetJointTargetPosition(jointshandle[i],jointscurrentpos[i])
            break
        end
    end
    while true do
        if (voltin==13) then
            break
        end
        out=simGetJointPosition(jointshandle[3])
        out2=out*180/math.pi
        simSetGraphUserData(graphHandle,'Simulation',out2)
        out=(volts[voltin]-1.37)*47.62+50
        simSetGraphUserData(graphHandle,'Robot',out)
        voltin=voltin+1
        break
    end
    cnt=cnt+simGetSimulationTimeStep()
    simAddStatusbarMessage('Simulation time: ' ..cnt)
end
if (sim_call_type==sim_childscriptcall_sensing) then
    -- Put your main SENSING code here
end
if (sim_call_type==sim_childscriptcall_cleanup) then
    -- Put some restoration code here
End
```

Source code in Arduino

```
#include <VarSpeedServo.h>

int l1=85; // variables for position

int l2=65;

int l3=90;
int l4=90;
int l5=90;
int r1=90;
int r2=110;
int r3=100;
int r4=90;
int r5=90;
int simpos[10];// positions for simulation
int simspeed[10];// speeds for each servo
int sensorValue=0;
int i=0;
float myvoltarray[13]; //array for saving voltages from servos
VarSpeedServo SerL1; ///servos obj
VarSpeedServo SerL2;
VarSpeedServo SerL3;
VarSpeedServo SerL4;
VarSpeedServo SerL5;
VarSpeedServo SerR1;
VarSpeedServo SerR2;
VarSpeedServo SerR3;
VarSpeedServo SerR4;
VarSpeedServo SerR5;
void setup()
{
  Serial.begin(38400);//open serial connection
  SerL1.attach(4); //left bottom second (near to floor)
  SerL2.attach(3);// left bottom first
  SerL3.attach(6);//left middle
  SerL4.attach(7);// left top second
  SerL5.attach(5);//left top first
  SerR1.attach(8);// right bottom second (near to floor)
  SerR2.attach(9);/// right bottom first
  SerR3.attach(10);// right middle
  SerR4.attach(12); // right top second
  SerR5.attach(11);//right top first
  SerL1.write(l1, 10);
  SerL2.write(l2, 10);
  SerL3.write(l3, 10);
  SerL4.write(l4, 10);
  SerL5.write(l5, 10);
  SerR1.write(r1, 10);
  SerR2.write(r2, 10);
  SerR3.write(r3, 10);
  SerR4.write(r4, 10);
  SerR5.write(r5, 10);
 delay(2000);
  for (i=1; i<14; i++) {
      myvoltarray[i]=0;
 }
  for (i=1; i<11; i++) {
      simspeed[i]=0;
      simpos[i]=0;
 }
}
void loop()
{
  l2=70;
  r2=108;
  l3=100;
  r3=90;
  simspos[3]=0; // values for simulation were set initially to avoid recalculations
  simspos[4]=0;
  simspos[8]=0;
  simspos[9]=0;
  simspeed[3]=6;
  simspeed[4]=5;
  simspeed[8]=6;
  simspeed[9]=5;
  for (i=1; i<11; i++) {
      Serial.print(simspos[i]);
      Serial.print(",");
  }
```

```
for (i=1; i<11; i++) {
    Serial.print(simspeed[i]);
    Serial.print(",");
}
for (i=1; i<13; i++) {
    Serial.print(myvoltarray[i]);
    Serial.print(",");
}
Serial.println(myvoltarray[13]);
SerL2.write(l2, 6);  // position, speed
SerR2.write(r2, 7);
SerL3.write(l3, 10);
SerR3.write(r3, 10);
sensorValue = analogRead(A4);
i=1;
for (i=1; i<14; i++) {
   sensorValue = analogRead(A4);
   myvoltarray[i] = sensorValue * (5.0 / 1023.0);
   delay(200);
}
l2=40;
r2=145;
l3=50;
r3=140;
simspos[3]=-50
simspos[4]=30
simspos[8]=-50
simspos[9]=30
simspeed[3]=5
simspeed[4]=6
simspeed[8]=5
simspeed[9]=6
for (i=1; i<11; i++) {
    Serial.print(simspos[i]);
    Serial.print(",");
}
```

```
  for (i=1; i<11; i++) {
      Serial.print(simspeed[i]);
      Serial.print(",");
  }
 for (i=1; i<13; i++) {
      Serial.print(myvoltarray[i]);
      Serial.print(",");
 }
 Serial.println(myvoltarray[13]);
  SerL2.write(l2, 6);
  SerR2.write(r2, 7);
  SerL3.write(l3, 10);
  SerR3.write(r3, 10);
  for (i=1; i<14; i++) {
     sensorValue = analogRead(A4);
     myvoltarray[i] = sensorValue * (5.0 / 1023.0);
     delay(200);
 }
}
```

APPENDIX B

Source code in V-REP

```
--DO NOT WRITE CODE OUTSIDE OF THE if-then-end SECTIONS
BELOW!!
--(unless the code is a function definition)
if (sim_call_type==sim_childscriptcall_initialization) then
    base=simGetObjectHandle('robot_dyn')
    graphHandle=simGetObjectHandle("graph1")
    graphHandle2=simGetObjectHandle("graph2")
     --defined all required arrays
    jointsname={} --Names
    jointshandle={} --Handles
    jointsposition={} -- current position
    readyjointsposition={} -- reculculated pos.
    minvolts={} -- array of min volts
    maxvolts={} -- array of max volts
    difference={} -- difference between positions in real
robot
    error1=0
    k=1
    for i=1,10,1 do
        jointsname[i]='joint_'..i
        jointshandle[i]=simGetObjectHandle(jointsname[i])
        jointsposition[i]=0
        readyjointsposition[i]=0

simSetJointTargetPosition(jointshandle[i],jointscurrentpos[i])
        minvolts[i]=0
        maxvolts[i]=0
        difference[i]=0
    end
    --- include the COM port and start of SERIAL connection
    portNumber="\\\\.\\COM3"
    --could be defined as followed
    --portNumber=[[\\.\COM12]]
    baudrate=250000
    serial=simSerialOpen(portNumber,baudrate)
    simAddStatusbarMessage('serial: '..serial)
    inpos=0
    k=1
    cnt=0
    volts={} --contains data from data stream
    for i=1,16,1 do
        volts[i]=0
    end
end
if (sim_call_type==sim_childscriptcall_actuation) then
```

```
    --get information from serail connection and separate of
them
    str=simSerialRead(serial,200,false,'\n',200)
    if str ~= nil then
        local token
        cpt=0
        --extracting the values in str separated by a ,
        for token in string.gmatch(str, "[^,]+") do
            cpt=cpt+1
            volts[cpt]=tonumber(token)
        end
        if cpt<15 then
            error=error+1
            if error>4 then
                while true do
                     simAddStatusbarMessage('Simulation failed
!')
                end
            end
        end
        if cpt>14 then
            error=0

simSetGraphUserData(graphHandle2,'Voltage',volts[4])
            for i=1,4,1 do
                jointsposition[i]=volts[i]
                minvolts[i]=volts[i+4]
                maxvolts[i]=volts[i+8]
                difference[i]=volts[i+12]
            end
            for i=1,4,1 do
                k=difference[i]/(maxvolts[i]-minvolts[i])
                simAddStatusbarMessage('volts: '..volts[i]..'
maxvolts:'..maxvolts[i]..' difference:'..difference[i])
                readyjointsposition[i]=(jointsposition[i]-
minvolts[i])*k*math.pi/180
                simAddStatusbarMessage('Data:
'..readyjointsposition[i]..'| Grad: '..jointsposition[i]..'|
k= '..k..'| min= '..minvolts[i]..'| max='..maxvolts[i]..'|
dif='..difference[i])
            end
            readyjointsposition[1]=-1*readyjointsposition[1]
            readyjointsposition[2]=-1*readyjointsposition[2]
            readyjointsposition[3]=1*readyjointsposition[3]
            readyjointsposition[4]=1*readyjointsposition[4]
            --set new position for all joints

simSetJointTargetPosition(jointshandle[8],readyjointsposition[
1])
```

```
simSetJointTargetPosition(jointshandle[3],readyjointsposition[
2])

simSetJointTargetPosition(jointshandle[9],readyjointsposition[
3])

simSetJointTargetPosition(jointshandle[4],readyjointsposition[
4])
            out=simGetJointPosition(jointshandle[3])
            out2=-1*out*180/math.pi
            --output data to graph
            simSetGraphUserData(graphHandle,'Simulation',out2)
            k=difference[2]/(maxvolts[2]-minvolts[2])
            out=(volts[2]-minvolts[2])*k
            simSetGraphUserData(graphHandle,'Robot',out)
        end
    end
    cnt=cnt+simGetSimulationTimeStep()
    simAddStatusbarMessage('Simulation time: ' ..cnt)
end
if (sim_call_type==sim_childscriptcall_sensing) then
    -- Put your main SENSING code here
end
if (sim_call_type==sim_childscriptcall_cleanup) then
    -- Put some restoration code here
end
```

Source code in Arduino

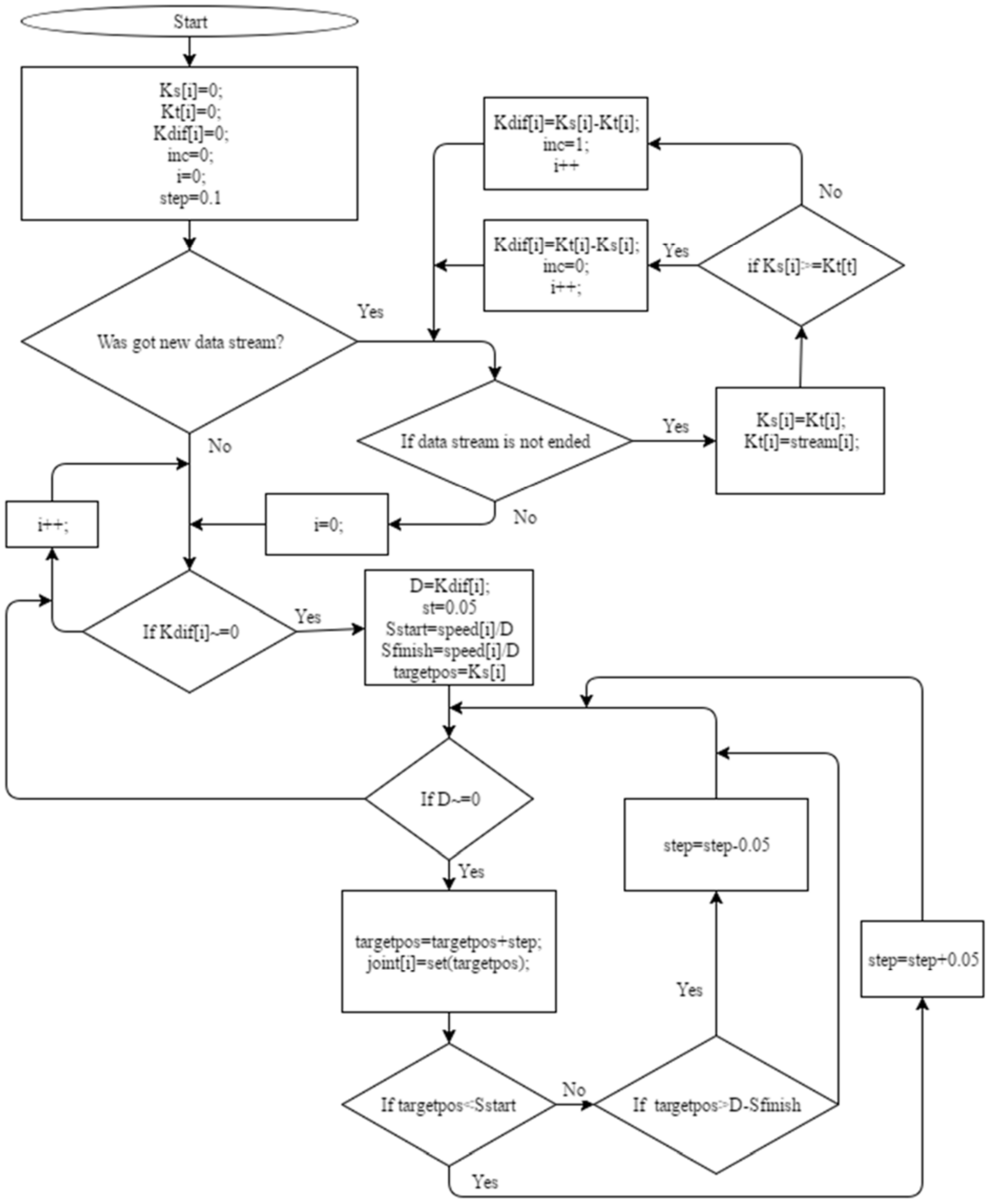
```
#include <VarSpeedServo.h>

//data: angles for V-REP

int l1=85; // bottom sec ////////////////////////////////////////k10

int l2=65; // BOTTOM FIRST // DECREASE TO SIT DOWN //////////////k9

int l3=90; // mid///////////////////////////////////////////////k8

int l4=90;

int l5=90;

int r1=90; //bottom sec//INSREASE TO BE CHANGED CM TO RIGHT//////k5

int r2=110;/////////////////////////////////////////////////////k4

int r3=100; //mid // INSREASE TO SIT DOWN///////////////////////k3

int r4=90;

int r5=90;

int k1=0;
int k2=0;
int k3=0;
int k4=0;
int ks1=0;
int ks2=0;
int ks3=0;
int ks4=0;
int sensorvalue[5][31]; // for sensor value
int i=0;//for iterations
int j=0; //for iterations
float myvoltarray[5]; //for voltages
float maxvolt[5]; //for max voltages
float minvolt[5];
int iterator=0;
///servos obj
VarSpeedServo SerL1;
VarSpeedServo SerL2;
VarSpeedServo SerL3;
VarSpeedServo SerL4;
VarSpeedServo SerL5;
VarSpeedServo SerR1;
VarSpeedServo SerR2;
VarSpeedServo SerR3;
VarSpeedServo SerR4;
VarSpeedServo SerR5;
void setup()
{
  Serial.begin(250000);//open serial connection
  SerL1.attach(4); //left botom second (near to floor)
  SerL2.attach(3);// left bottom first
  SerL3.attach(6);//left middle
  SerL4.attach(7);// left top second
  SerL5.attach(5);//left top first
  SerR1.attach(8);// right bottom second (near to floor)
  SerR2.attach(9);/// right bottom first
  SerR3.attach(10);// right middle
  SerR4.attach(12); // right top second
  SerR5.attach(11);//right top first

  SerL1.write(l1, 10);
  SerL2.write(l2, 10);
  SerL3.write(l3, 10);
  SerL4.write(l4, 10);
  SerL5.write(l5, 10);
  SerR1.write(r1, 10);
  SerR2.write(r2, 10);
  SerR3.write(r3, 10);
  SerR4.write(r4, 10);
  SerR5.write(r5, 10);
  delay(2000);
}
void loop()
{
  if (iterator < 1) {
    sensorvalue[1][1] = analogRead(A5);//right mid
    sensorvalue[2][1] = analogRead(A4);//left mid
    sensorvalue[3][1] = analogRead(A3);//right bottom
    sensorvalue[4][1] = analogRead(A2);//left bottom
    myvoltarray[1]=sensorvalue[1][i] * (5.0 / 1023.0);
    myvoltarray[2]=(1023.0-sensorvalue[2][i]) * (5.0 / 1023.0);
    myvoltarray[3]=sensorvalue[3][i] * (5.0 / 1023.0);
    myvoltarray[4]=(1023.0-sensorvalue[4][i]) * (5.0 / 1023.0);
    for (i=1; i<5; i++) {
      //determine some existing value for each servo just in first iteration
      minvolt[i] = sensorvalue[i][1] * (5.0 / 1023.0);
      maxvolt[i] = sensorvalue[i][1] * (5.0 / 1023.0);
    }
  }
  l2=65;
  r2=120;
  l3=100;
  r3=90;
  if (iterator < 1) {
    ks1=r3;
    ks2=l3;
    ks3=r2;
    ks4=l2;
  }
  SerL2.write(l2, 7);  // position, speed
  SerR2.write(r2, 7);
  SerL3.write(l3, 10);
  SerR3.write(r3, 10);
  for (i=1; i<16; i++) {
      sensorvalue[1][i] = analogRead(A5);//right mid
      sensorvalue[2][i] = analogRead(A4);//right mid
      sensorvalue[3][i] = analogRead(A3);//right mid
      sensorvalue[4][i] = analogRead(A2);//right mid
      myvoltarray[1]=sensorvalue[1][i] * (5.0 / 1023.0);
      myvoltarray[2]=(1023.0-sensorvalue[2][i]) * (5.0 / 1023.0);
      myvoltarray[3]=sensorvalue[3][i] * (5.0 / 1023.0);
      myvoltarray[4]=(1023.0-sensorvalue[4][i]) * (5.0 / 1023.0);
      if (iterator < 1) {
        for (j=1; j<5; j++) {
          if (myvoltarray[j] < minvolt[j]){
            minvolt[j]=myvoltarray[j];
          }
          if (myvoltarray[j] > maxvolt[j]){
            maxvolt[j]=myvoltarray[j];
          }
        }
      }
      if (iterator > 0){
        for (j=1; j<5; j++) {
           Serial.print(myvoltarray[j]);
           Serial.print(",");
        }
        for (j=1; j<5; j++) {
           Serial.print(minvolt[j]);
```

```
         Serial.print(",");
      }
      for (j=1; j<5; j++) {
         Serial.print(maxvolt[j]);
         Serial.print(",");
      }
      Serial.print(k1);
      Serial.print(",");
      Serial.print(k2);
      Serial.print(",");
      Serial.print(k3);
      Serial.print(",");
      Serial.print(k4);
      Serial.print(",");
      Serial.println(k4);
    }
    delay(250);
 }
   //---------------------------------------------------
l2=45;
r2=140;
l3=60;
r3=130;
if (iterator < 1) {
  k1=r3-ks1;
  k2=ks2-l3;
  k3=r2-ks3;
  k4=ks4-l2;
}
SerL2.write(l2, 6);
SerR2.write(r2, 7);
SerL3.write(l3, 10);
SerR3.write(r3, 10);
for (i=1; i<16; i++) {
    sensorvalue[1][i] = analogRead(A5);//right mid
    sensorvalue[2][i] = analogRead(A4);//right mid
    sensorvalue[3][i] = analogRead(A3);//right mid
    sensorvalue[4][i] = analogRead(A2);//right mid
    myvoltarray[1]=sensorvalue[1][i] * (5.0 / 1023.0);
    myvoltarray[2]=(1023.0-sensorvalue[2][i]) * (5.0 / 1023.0);
    myvoltarray[3]=sensorvalue[3][i] * (5.0 / 1023.0);
    myvoltarray[4]=(1023.0-sensorvalue[4][i]) * (5.0 / 1023.0);
    if (iterator < 1) {
        for (j=1; j<5; j++) {
          if (myvoltarray[j] < minvolt[j]){
            minvolt[j]=myvoltarray[j];
          }
          if (myvoltarray[j] > maxvolt[j]){
            maxvolt[j]=myvoltarray[j];
          }
        }
    }
    if (iterator > 0){
        for (j=1; j<5; j++) {
          Serial.print(myvoltarray[j]);
          Serial.print(",");
        }
        for (j=1; j<5; j++) {
          Serial.print(minvolt[j]);
          Serial.print(",");
        }
        for (j=1; j<5; j++) {
          Serial.print(maxvolt[j]);
          Serial.print(",");
        }
        Serial.print(k1);
        Serial.print(",");
        Serial.print(k2);
        Serial.print(",");
        Serial.print(k3);
        Serial.print(",");
        Serial.print(k4);
```

```
          Serial.print(",");
          Serial.println(k4);
      }
      delay(250);
    }
  iterator=1;
}
```

APPENDIX C

Full block diagram for “pre targeting” mode


Start
Ks[i]=0;
Kt[i]=0;
Kdif[i]=0;
inc=0;
i=0;
step=0.1
Kdif[i]=Ks[i]-Kt[i];
inc=1;
i++
No
Kdif[i]=Kt[i]-Ks[i];
inc=0;
i++;
Yes
if Ks[i]>=Kt[t]
Yes
Was got new data stream?
No
If data stream is not ended
Yes
Ks[i]=Kt[i];
Kt[i]=stream[i];
i++;
i=0;
No
If Kdif[i]~=0
Yes
D=Kdif[i];
st=0.05
Sstart=speed[i]/D
Sfinish=speed[i]/D
targetpos=Ks[i]
If D~=0
Yes
step=step-0.05
step=step+0.05
targetpos=targetpos+step;
joint[i]=set(targetpos);
Yes
If targetpos<Sstart
No
If targetpos>D-Sfinish
Yes