BIOS4YOU

Bio-Inspired STEM topics for engaging young generations

project number: 2019-1-DE03-KA201-060125

KTU Learning Unit:
Robotics in Biotechnology

Karlsruher Institut für Technologie

kauno technologijos universitetas

UNIVERSITÀ DEGLI STUDI DI PALERMO

Co-funded by the Erasmus+ Programme of the European Union

This project has been funded with support from the European Commission - Application number 2019-1-DE03-KA201-060125.
This publication [communication] reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Table of Contents

Card of the learning path

General topic of the learning path	*Robotics in Biotechnology*
Specific name of the learning unit	*Building the Bionic Hand*
Target user age	*17-19 years*
Learner prerequisites	*Basic knowledge of Physics, Mathematics, IT, Biology*
Description of the learning unit	*The final practical result – to make a part of a bionic hand prototype. Here* teachers and students will be *acquainted with bionic manipulators principles, with the 3D printing and 3D modeling techniques, how to visually/graphically program a microcontroller and make the bionic fingers move depending on the buttons pushed by the user.* *Practical exercises involve: Tinkercad 3D graphics modeling; Participation in the 3D printing procedure; Practical programming and experimentation with the constructed bionic hand parts.*
Subject involved	*Mathematics, Physics, Information technology, Biology*
Keywords	*Bionics, 3D modeling, 3D printing, Arduino, Robotics*
Key-skills, abilities, knowledge that can be acquired	*Problem-solving, design thinking. Knowledge about bionic and robotic manipulators, knowledge about 3D printing principles. Knowledge about different 3D modeling software types. Skils of 3D modeling and visual programming of AVR controllers.*
Resources and didactic tools used	*Web-based resources, research papers,*
Evaluation criteria and assessment	*Different levels of concept knowledge, abilities and skills achieved*

1. Bionic Hand – an Overview

The first general impression about robots – this is some artificial machine, similar to a human body, and acting similar to us. We even expect talking to us and understanding us. Therefore, such robots are called humanoids. Actually, there are lots of much simpler mechanical robots used in industry. They usually perform only one or some actions (e.g. transferring an item being manufactured from one worker to another), but general principles of operation are the same.

Figure 1. Example of a Bionic Hand

The situation does not look so complicated, if broken down into parts and looked from inside – recent technological achievements allow us to create a bionic hand ourselves (DIY = "do it yourself"):

- The "body" parts of a bionic hand could be produced by the use of a 3D printer;
- There should be some space for mechanical and electronic components inside it;
- These "body" parts are usually designed by employing 3D modeling software;
- The motions are realized by small motors ("muscles"), pulling lines/strings ("tendons") attached to proper parts of the built hand;
- The motors are driven from a software program, that runs on the microcontroller.

Fiure 2. Robot humanoid

- The motions could be initiated by electric pulses existing inside a human body – these pulses are detected and transferred to the microcontroller.

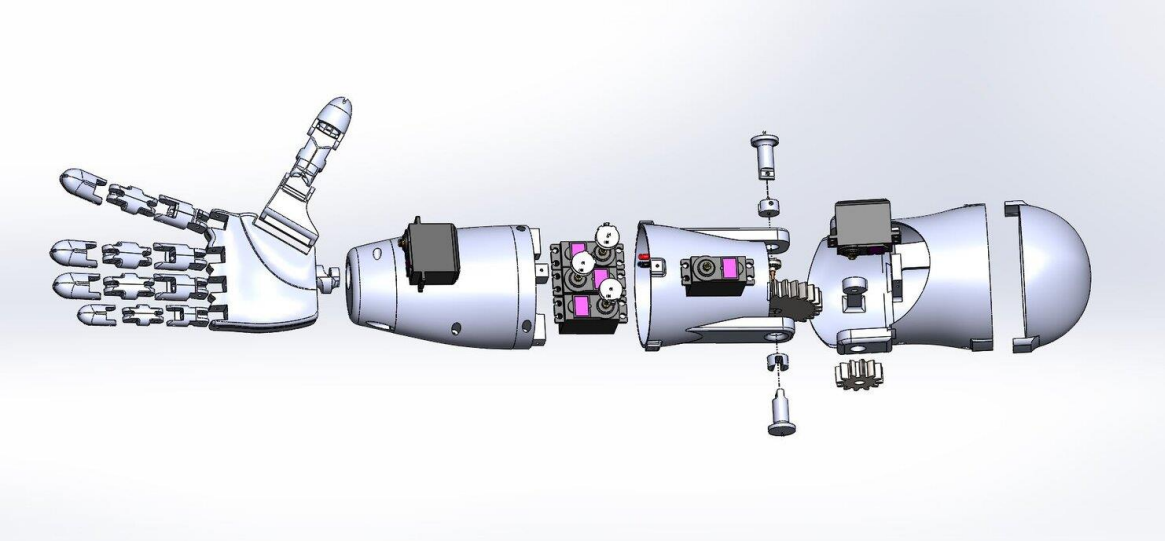


Figure 3. Main parts and components of a bionic hand

The following topics will help us to better understand how these steps are realized, and even to try them ourselves.

Modelling and design

Initially, we shall limit ourselves to simpler movements – only a rotation between two adjacent finger parts. There should be enough space without obstacles for moving strings inside the body, and an opportunity to install them (any internal parts) conveniently. Therefore, “a body part” is expected to be composed from two pieces, having axis holes and space shifts for joining different “body parts” altogether.

The initial shape for modelling fingers are half-cut cylinders. We need parts to overlap partially in order to be fixed one to each other or make a rotating connection.

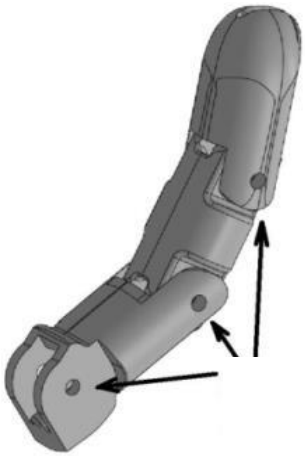


Figure 4. Composition of a finger

The space inside fingers is not enough for motors to be fitted inside (or such micro-motors might cost too much), therefore finger parts are moved by pulling strings, and the motors are usually placed in the wrist or the arm part of a bionic hand.

So, initially, we need to manufacture all necessary parts of a bionic hand. For this, the first step is to create models of these parts. And after that, the obtained models are used to generate instruction files for 3D printers which help us in the manufacturing process.

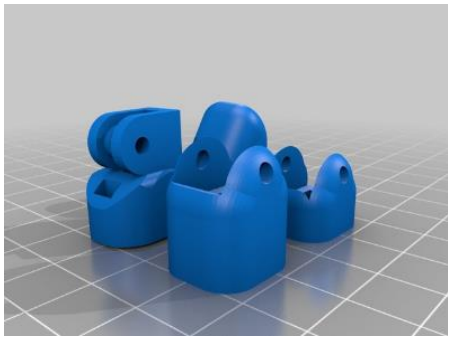


Figure 5. Separated parts of a finger

3D printing

In general, 3D printing technology allows us to create an object of almost any shape. The technology evolved from ink-jet printing: melted plastic (instead of ink) is sprayed layer by layer according to the instruction file and becomes solid at room temperature. Besides the mentioned way of 3D printing, there are several other technologies that allow customers to produce parts of different materials such as metal, concrete, rubber, etc.

For example, in the special 3D printing technique to produce metal parts the metal object is generated in the sand dust; the metal is melted into a solid object heated by a laser.

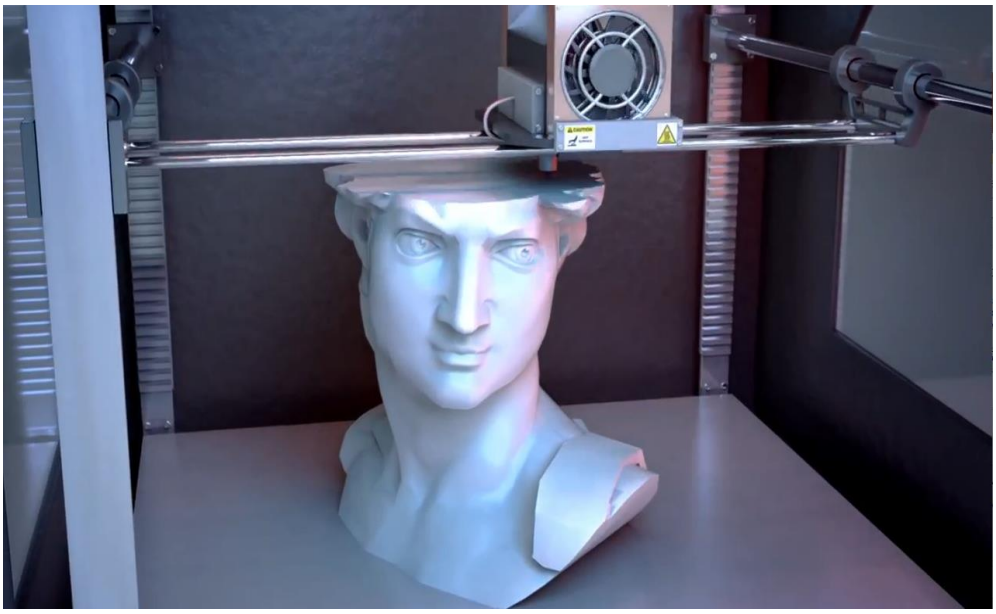


Figure 6. 3D printing technology enables to create object of almost any shape

"Muscles" pulling "tendons"

The movement of a robotic hand is very similar to that of a human hand. The muscle contracts when an electric current passes through it[1]. It pulls tendons attached to an adjacent body part. That rotates around a joint connecting these two body parts.

In a case of robot, it looks similar – a motor pulls a string. In order to deploy a reverse movement, two alternatives can be used:

- a spring, forcing a body part to an initial position;
- two-string system, pulling the body part both directions.

In any of the cases, the motion is realized by special kind of motors, called servo-motors (or servo-actuators). These digital devices move to and stop only in certain positions, - differently to all-time rotating motors.

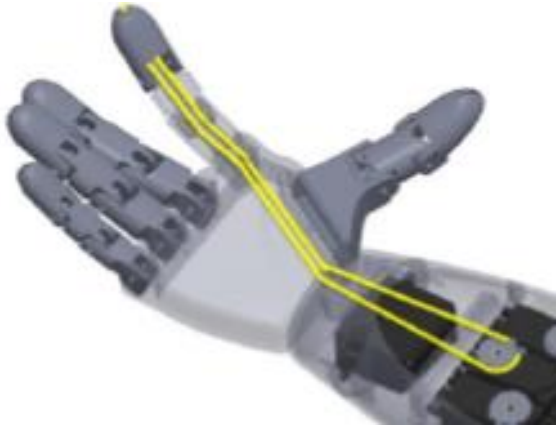


Figure 7. Mechanics, that moves a finger of a bionic hand

Programming and controlling the motion

The target position for motors of a bionic hand is set by a microcontroller – a small "computer" with a software inside.

There are several important things how the software controls the motion:

- the program is repeated almost endlessly (e.g. until a special condition is met or power supply is switched off);
- the program checks the mio-sensor (a muscle sensor) to know when to start pulling string (and which one);
- if we know in advance what object we would like to take by our bionic hand, then we can instruct the motor about the target position;

Figure 8. Bionic hand holding an object

[1] The muscle contractions are controlled by very small electrical pulses, that might be measured by special sensors. These pulses are comming from brain and might be used to contol a bionic hand, attached as a replacement of one, lost during an accident.

- In most of cases the shape of an object is not known, then we need another sensor at the end of a finger to allow force be checked, not to squeeze the object in our bionic hand.

Figure 9. Force sensors placed at fingers' end

2. 3D printing principles and 3D modelling

3D printers

3D printing is the process of turning a three-dimensional computer model into a real thing. It is a fairly new technology that is evolving rapidly and is surprising in its capabilities and adaptability. If until very recently 3D printing was a real breakthrough, now these technologies are being applied in an increasingly diverse area of life. Not only individual parts are printed, but also machine or house constructions. Increasingly, 3D printing is being used in medicine. Teeth, jaws and other bones or parts thereof printing and reproduction is no longer the latest discovery. Researchers are actively working to print and use human skin tissue and the heart. This will avoid waiting for the donor organ, reduce the risk of organ rejection or infection, and reduce other factors.

The material used for printing varies depending on the purpose of the model. These materials are evolving along with the technology itself. Simple 3D printing takes place by melting a plastic spool. Carbon fiber is used to print patterns that need to be extremely light but durable. Other materials used in the machine or home industry, which are subject to particularly high requirements for durability, moisture, etc. In medicine, the innovator in printing human organs was the Swedish company Cellink, which invented the printing material for human organs.

Figure 10. Printing car parts with a 3D printer

Figure 11. House construction printing with a 3D printer

Figure 12. Organ printing with a 3D printer

3D Printing Process

The printing process can be represented by the sequence shown in Figure 13:

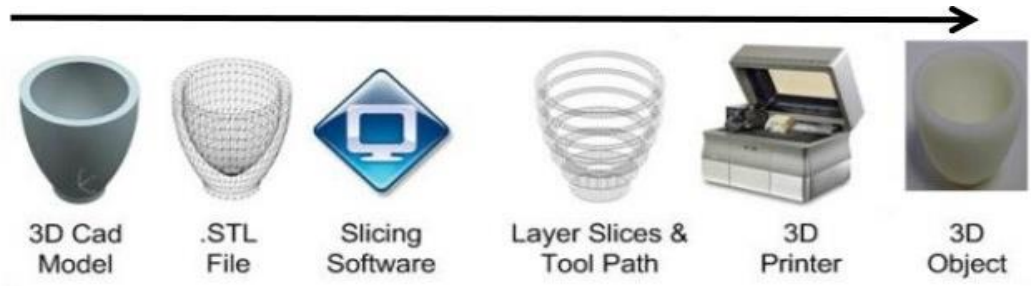


Figure 13. 3D printing process

Once the model has been created by the modelling program, it must be saved in *.STL format. This format is for 3D printing. In this way, the program recognizes the information in the file and divides

the model itself into layers. The 3D printer starts working from the lowest layer, filling in all the information. It then moves to the next layer until all the layers are printed vertically.

It is necessary to know the basic parts of a 3D printer to understand the printing process.

There may be various printer modifications depending on the needs, type, price and other criteria. However, the basic parts remain the same. By default, such a printer consists of a spool of material (1). It can be plastic, metal, glass or others. Also the print head (2) moving transversely and longitudinally. The printable height of the model is adjusted by a sliding background (Figure 14).

Figure 14. 3D printer

When the model is divided into layers, printing starts from the lowest layer. The print head (Figure 15a), after melting the feed material (1) from the spool, prints one layer horizontally (Figure 15b). Then move on to the next one.

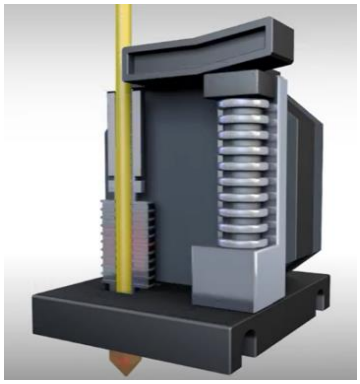


Figure 15a. Printhead

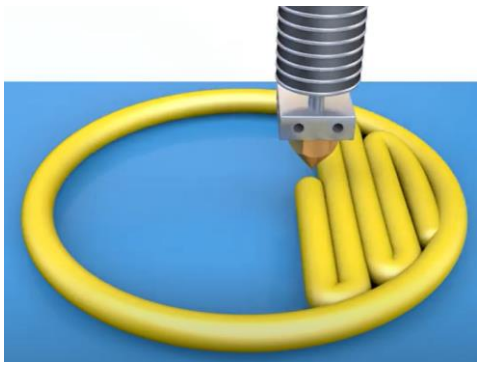


Figure 15b. One layer is printed

3D modelling

In order to transmit the information to the printer, the item must be modelled at first. It is important to notice that the printer "reads" the information consistently, not in the way we see it. That is, if during modelling, we draw a line on a line, we see only one line, and the printer prints the first line first, then the second line on the same layer. This increases material costs and the actual model may be inaccurate. Therefore, in modelling, it is necessary to make sure that there are no duplicates in the layer, that the line layer thickness and other parameters are properly provided.

The choice of a 3D modelling software depends on many factors. This includes user computer literacy, ability to manage a simulation program, your computer's computational capabilities, project complexity, software license, etc. Licensed and open source programs can be distinguished according to the "availability" of the software. Licensed have far more options. Open

source programs are available to every user, are versatile enough for the operating systems, but are more likely to have some “bug” in the software itself.

The most commonly used modelling software is shown in the table below:

Proprietary	Open source
AutoCad	Blender
Fusion 360	TinkerCad

One of the largest companies involved in modelling software development is Autodesk. It includes AutoCad, Fusion 360, as well as TinkerCad software packages. Tinkercad is available as a free online collection of software tools set which is very handy for usage in schools o for other teaching purposes. Autodesk provides services for both the business classroom and learning purposes. Educational institutions can get a license for products. It is not a complete software package that is paid and intended for the business class, but to study, model and prepare a model for 3D printing is fully sufficient.

Each version of the software is constantly updated and supplemented with additional features. The main goals are to create intuitive software that is as easy to learn as possible for the user. Each of them requires a basic theoretical basis for modelling.

3. Modelling software overview

AutoCAD software

The program is designed for modelling two-dimensional and three-dimensional drawings, diagrams and images. There are numerous environments implemented in the software. These environments differ only in the tools icons in the menu bar. If you know name of the function, it doesn't matter in witch environment you are working. The program communicates with the user via the command line. Starting from version 2018, for the convenience of the user, it is enough the cursor to be in the program window, when you start typing the name of the function, the program "offers" possible commands for the manipulation. Big advantage of this program is easy generation of projections from a three-dimensional model. Once a model has been created, a material or group of materials can be assigned to it. This program is not very suitable for visualization, because in order to get the most realistic image, it takes many computer resources for calculations – to adjust lighting, materials and other parameters.

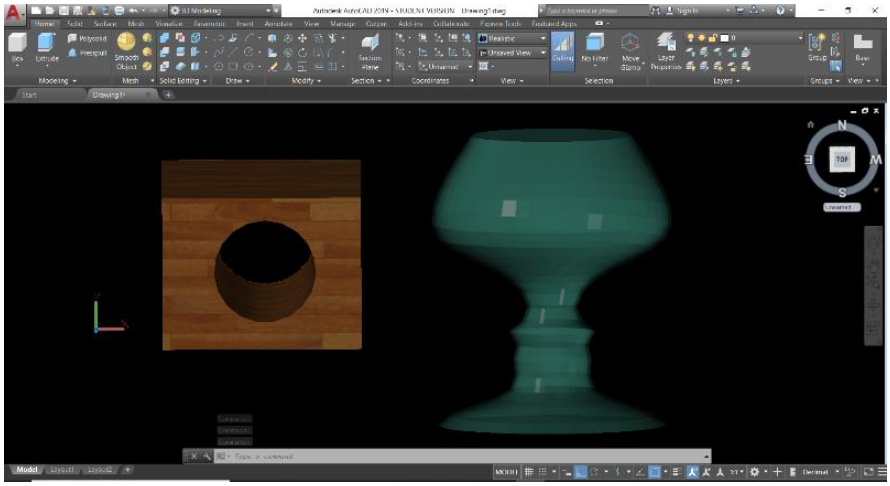


Figure 16. AutoCad application environment

Fusion 360 software

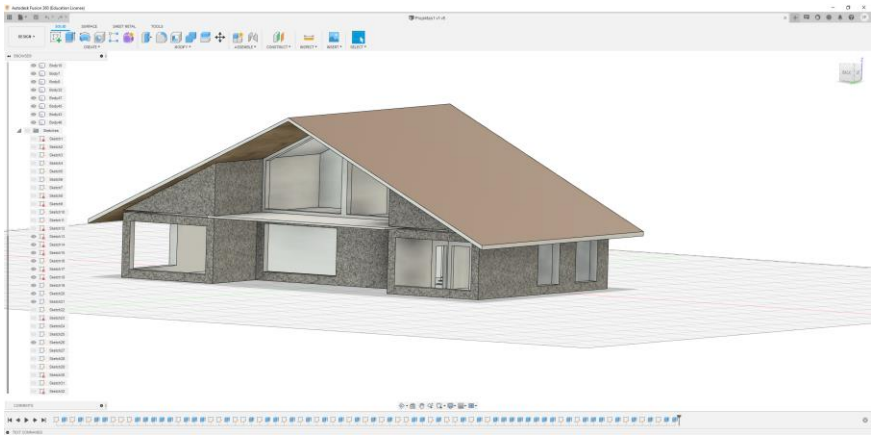


Figure 17. Fusion 360 Application environment

Fusion 360 is a simpler, less skilful program. Having all the same tools as AutoCad. One of the advantages is easier material collection. The created model is easy to upload to a photo of the real environment, viewed from all sides.

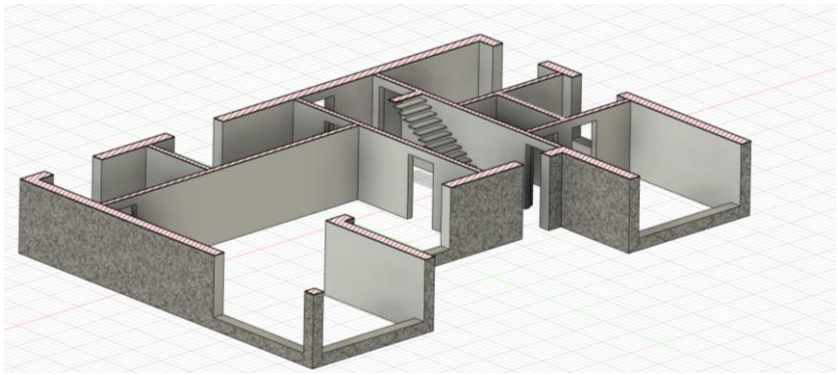


Figure 18. View the model by "hiding" part of the objects

Blender software

An open source program for 3D modelling and animation. The main advantage of this software that it is free and easy to install. The disadvantage – it is quite complex to work with, requiring more skills and experience. The icons are not arranged in the menu bar. Knowing the name of a function makes it easy to call it through a search. This program has most of the benefits of the programs listed above. The modelling can be performed in more than one method. One way is to modify three-dimensional primitives, such as cube, cylinder, sphere. The feature of the program is the possibility of animation. It is possible not only to design a model, but also to see how it can actually move. For example, when constructing a hand, the app allows you to add restrictions so that your fingers can only bend to a predetermined angle, not 360 degrees. Another advantage is the material settings. Program focuses on 3D animation, silky material, hair, water, and other materials can be extracted for realistic images. Like other simulation programs, it has the ability to export the file to *.stl format, ready for 3D printing.

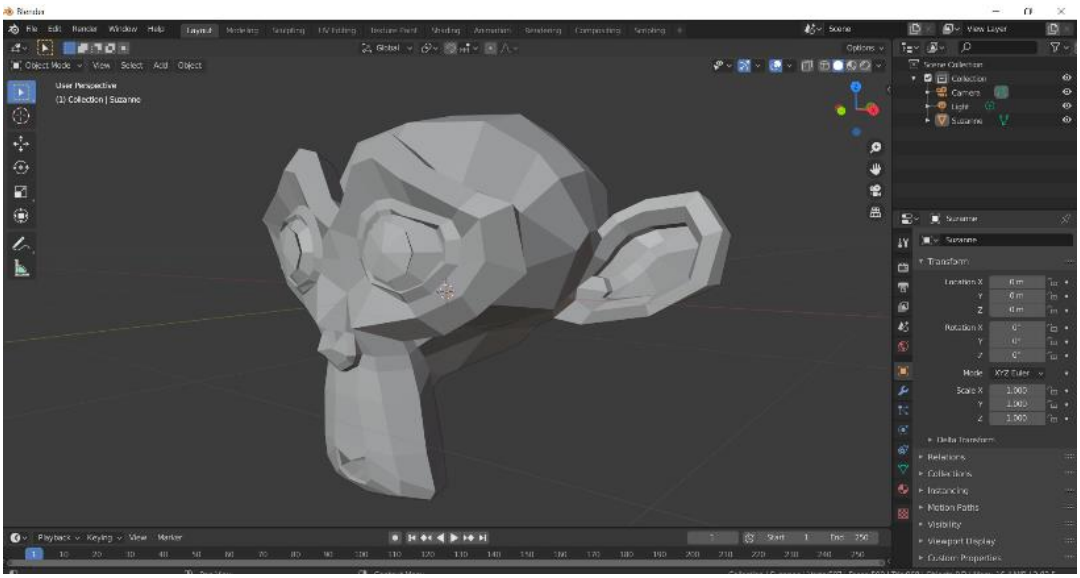


Figure 191. Blender software environment

TinkerCad software

One of the main advantages over the programs listed above is the ability to work online. This is an Autodesk software package that can offer many additional tools with a license. However, for small projects, the tools offered by manufacturers in the free online version are fully sufficient.

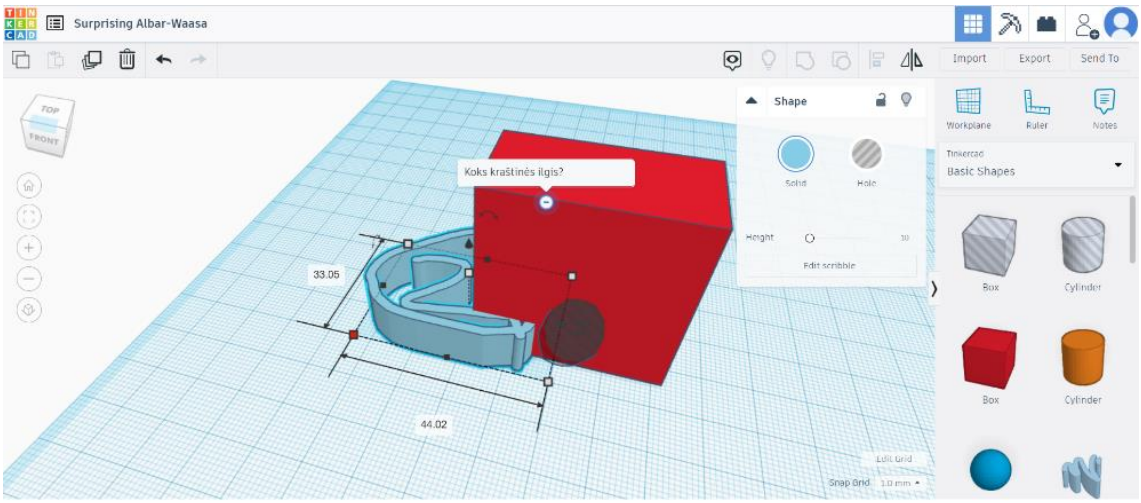


Figure 20. TinkerCad application environment

The program is especially suitable for children because all the tools are visible and clearly understood. This online version has advantages. The teacher can create classes and teach for the whole class. Projects can also be done in teams by "inviting" another user to the project. Another very handy tool is comments. They can be both written and hidden. The program allows you to import already created *.STL files and edit them. There may be problems editing individual blocks in the model because such blocks are "locked" by grouped parts.

Which program to choose for modelling, depends primarily on the goal and the user's skills.

Related links:

1. 3D printing parts of car: https://blogs.3ds.com/3dsmobility/wp-content/uploads/sites/13/2017/04/Additive-Manufacturing.jpg
2. 3D printing process: https://www.youtube.com/watch?v=qoBU0r7pT84
3. Steps of 3D printing: https://upload.wikimedia.org/wikipedia/commons/2/2c/3d-printing-a-2014-horizonwatching-trend-summary-report-9-638.jpg
4. 3D printer: https://www.cleanpng.com/png-3d-printing-filament-3d-printers-aleph-objects-inc-1348799/preview.html

4. Modeling concept and basic manipulations

Before to start modeling, we need to understand the basic principles and differences between three-dimensional and two-dimensional modeling.

Two-dimensional modeling is working in a plane. In other words, we model on a sheet of paper where we can draw across and along the sheet. In the design language, this is horizontally or vertically, or in the x and y axis directions

Three-dimensional modeling differs from two-dimensional by one more axis, which usually indicates the height of the model. Thus, three-dimensional models are designed on the x, y, and z axes.

Creating a model, it is not enough to imagine what it should look like. Its specific dimensions are needed. When creating a three-dimensional model, measurements are required with respect to the x, y, and z axes. In other words, a description of the model with x, y, and z projections is required. The projection is a two-dimensional plane aligned perpendicular to the two axes, the direction of the third axis being perpendicular to us. Another important highlight is the marking of visible and invisible lines. The visible line is the kind of edge that we see from the observer's point of view, it is not covered by anyone. The invisible line (red in Figure 1) is the edge which, when viewed from the observer's point, is covered by a plane closer to the observer.

Projections are an agreement document between engineers with not only drawings but also dimensions on them. With such dimensional drawings, it is easy to create a three-dimensional model with any program that is convenient for us. For example, you need to model a box of certain dimensions with a cylindrical hole. In order to model, we first need projections with dimensions. Only then it can be modeled accurately.

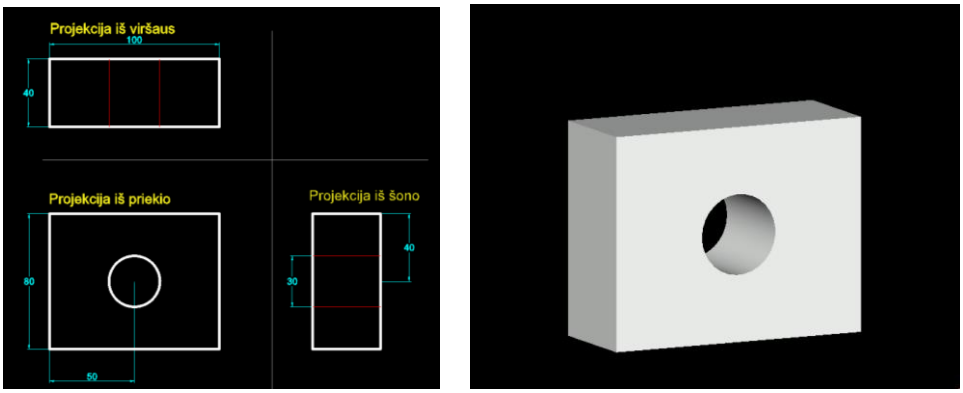


Figure 21. Detail projection with dimensions and its three-dimensional model

Simulation of TinkerCad program

To understand the simulation process, let's try the open source virtual program TinkerCad.

1. Click on the link https://www.tinkercad.com/dashboard. Clicking "Creat new design" prepares the desktop.
2. Drag the cube into the empty space from the toolbar on the right. On the left side of the window, the navigation tool (cube-shaped) must be set to "Front".

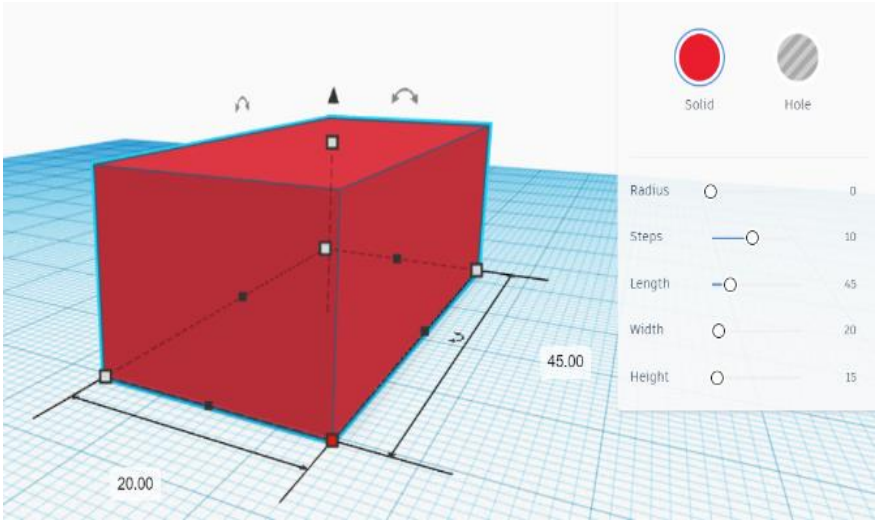


Figure 2. Editing parameters of cube

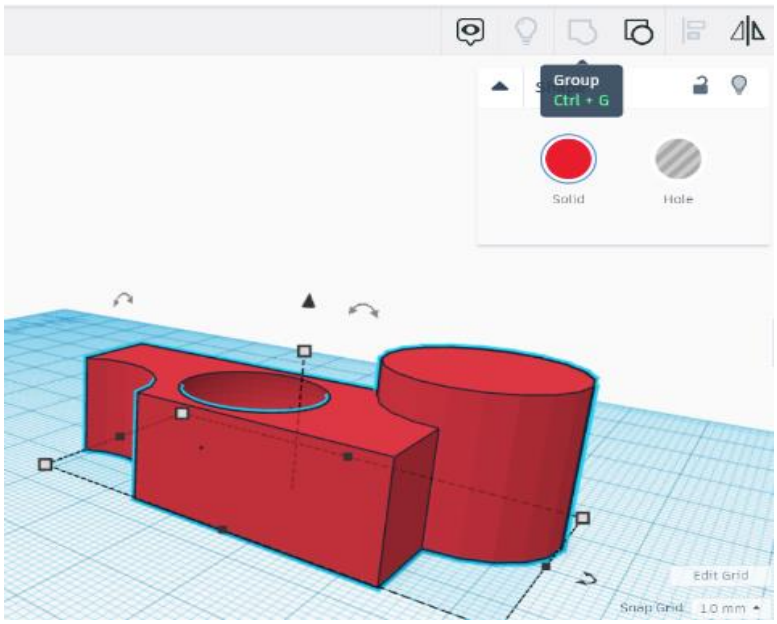


Figure 3. Scene view after grouping

3. To change the parameters of a cube, clicking on it opens a window with the parameters next to it (Figure 22).
4. The program allows you to choose from different types of shapes. Possible settings are a convex model or a recess (or hole) in its shape.
5. Select the sphere tool, load it in the top plane of the box, and set the hole parameter. This will give us a spherical recess in the cube.
6. Place two cylinders on the different edges of the box. Set the property to "solid" for one and "hole" for the other. The shapes assigned to the "hole" feature look like glass. But this is

only an imaginary part of that figure. In order to "cut" holes in the box, all parts of the scene need to be grouped.

7. Select all objects in the scene and select the "group" function (Figure 3).
8. All you have to do is export the prepared model to * .stl file (Figure 4). You can choose to print the entire model or specify specific parts.
9. There is also the option to save the model, or print directly. TinkerCad has the ability not only to print the model, but also to cut it with a laser machine.

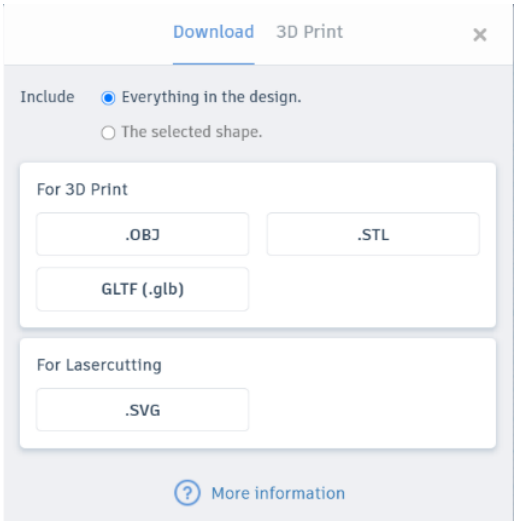


Figure 4. Exporting a file

Creating an accurate model

If you notice, the TinkerCad workspace has a horizontal platform on which we design the model. It is possible to change the grid of this base, its step. The step of movement of the part will depend on it (Figure 25).

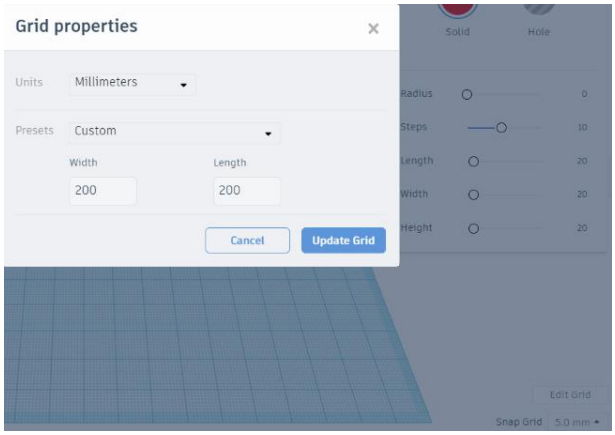


Figure 5. Change the grid step

The grid does not have an absolute coordinate system. That is, where you placed the shape, the program calculates the start of the coordinates. Dragging the object, the deviation is calculated relative to the previous location.

According to the modeling rules, the model is projected into an XY plane that coincides with the ground. To project in a different direction, you must first change the working plane.

Let's try to design the model of Figure 26 (1).

1. Let's drag the cube into the working field and set its dimensions 40x100x80
2. Also drag cylinder to the scene. If you notice, the background of the cylinder coincides with the working plane. We can solve this problem in two ways:
    - Change the working environment to the side of the box
    - Rotate the cylinder 90 degrees to the x axis

Changing the working plane. Select the "workplane" icon (Fig. 26 (1)) and click on the box wall (Fig. 26 (2)). Now that the cylinder is pulled, its base will coincide with the new working plane, which means with the side plane of the box.

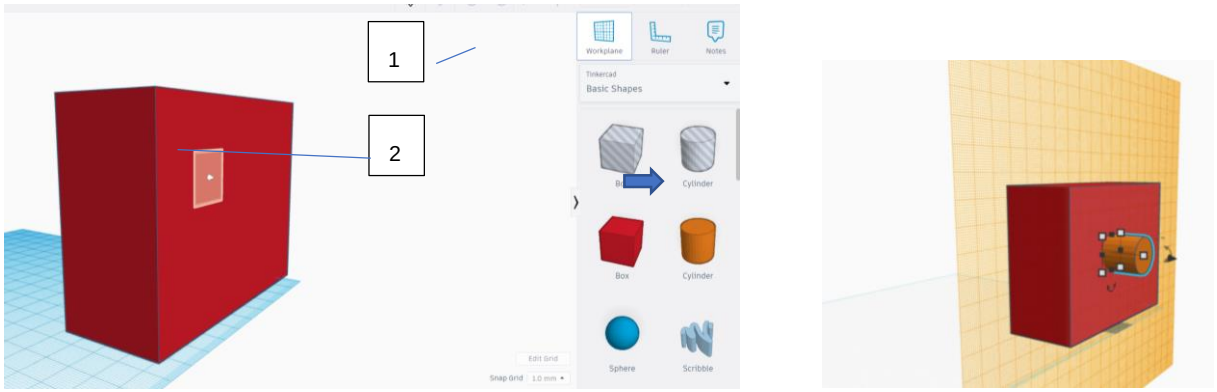


Figure 6. Changing the working plane

Cylinder rotation. Drag cylinder and select it. Click on the arrows showing the rotation with respect to the x-axis and set the rotation angle to 90 degrees.

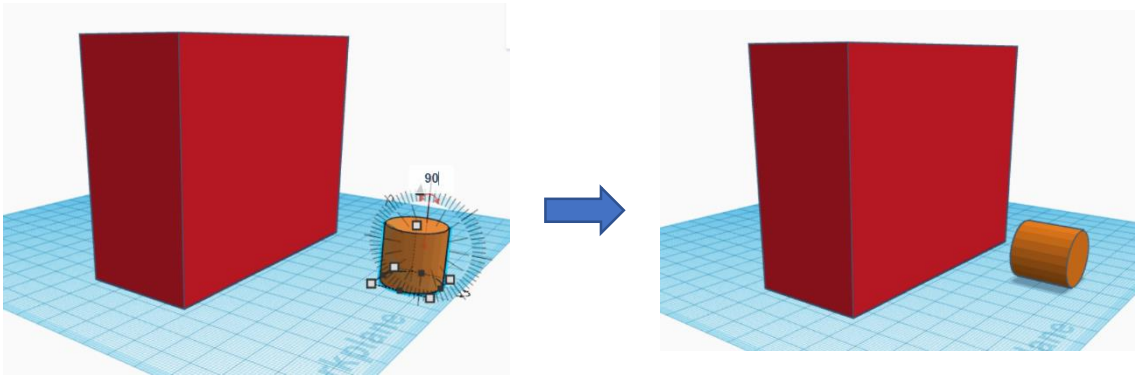


Figure 7. Rotation of the cylinder with respect to the x axis

3. Location of cylinder hole. For moving cylinder to special location, we must change working plane to the side of the box. So it is better to change it before including cylinder shape.
    - Align both shape by one edge.
    - Move cylinder 35 mm to the right and 25mm to the top
    - Change working plane to the ground and set navigation bar to TOP
    - Move cylinder vertically and fit to the box (Figure 28)
4. Now as cylinder is in correct place, we can change type of shape to “hole”
5. Select both shapes, set “group”
6. Export file to *.stl

Figure 9 shows how final model looks in TinkerCad program (left) and as *.STL file (right)

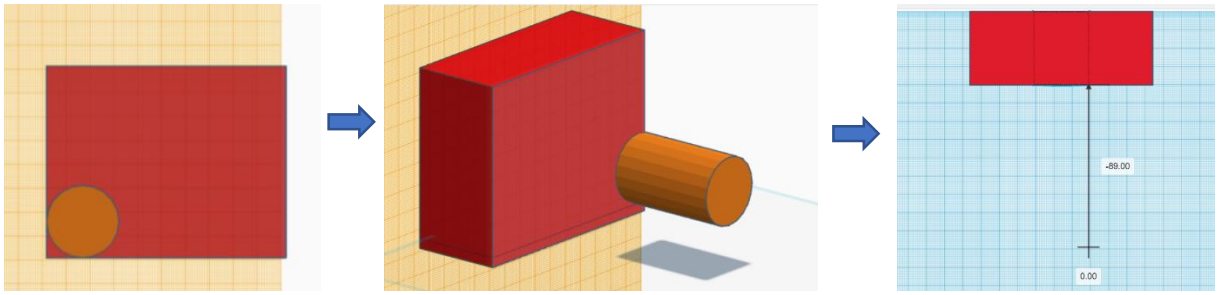


Figure 28. Moving cylinder

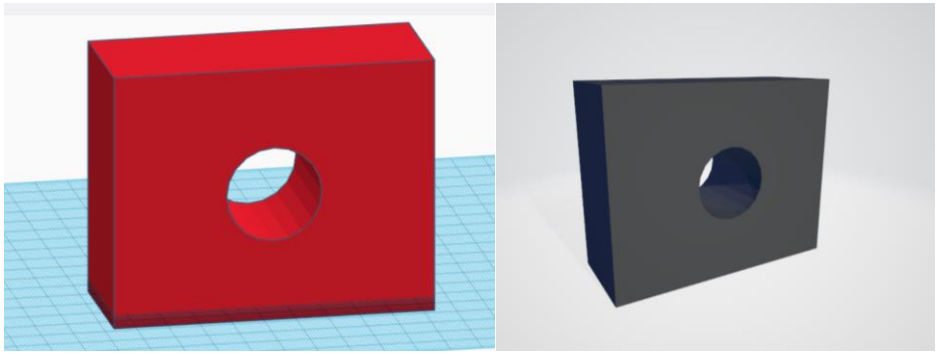


*Figure 8. Final model in TinkerCad environment and *.stl file*

TinkerCad is convenient to try modeling tool for beginners. For complex projects it is better to choose Blender program.

Short review, how this simple model could be created using blender. Default view of starting program is in image 30. Default shape is cube, units – meters, length 2m.

Steps to begin:

1. Change units to millimeters
2. Change length of edges to x, y and z axis 40x100x80
3. Select shape and press CTRL+A and select **Scale.** With this operation we set new shape of box as default.

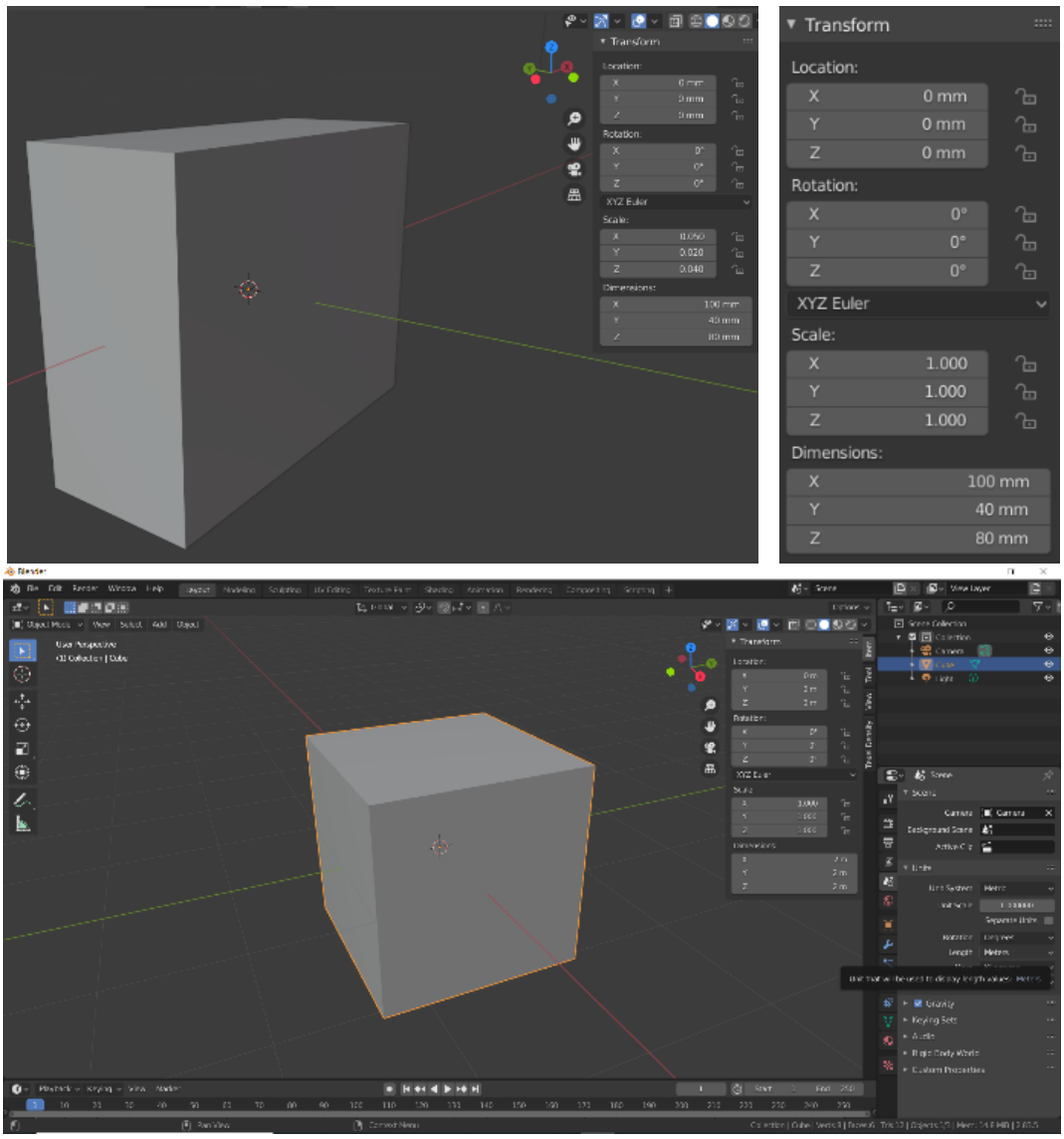


Figure 9. Box transformation

4. Press **Tab** to switch to *Edit Mode*
5. Press SHIFT+A and select Cylinder shape.
6. This shape by default is oriented by Z axis(Figure 30). So we need to change options of it:
    - Radius of background is 15mm
    - Rotate by Y axis 90 degrees

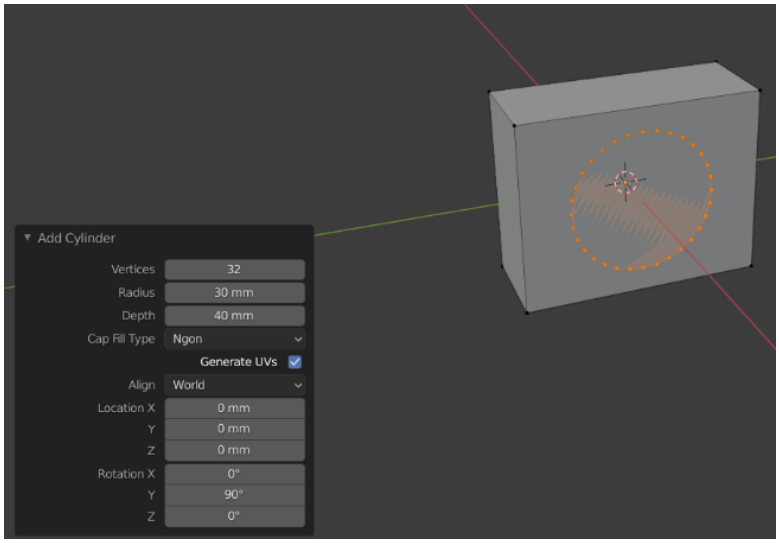


Figure 31. Cylinder options

7. Now we have two duplicated planes – box and cylinder background. Delete theses planes (two backgrounds of box and two of cylinder).
8. To make a plane between edges of box and cylinder auxiliary edges are needed. It is simple to calculate distance between point of circles diameter and box edge horizontally. It is 35mm. Select circles diameters point, press E (extrude), lock Y axis and set 35 (Figure 12). At the top left corner you can check what options are set.

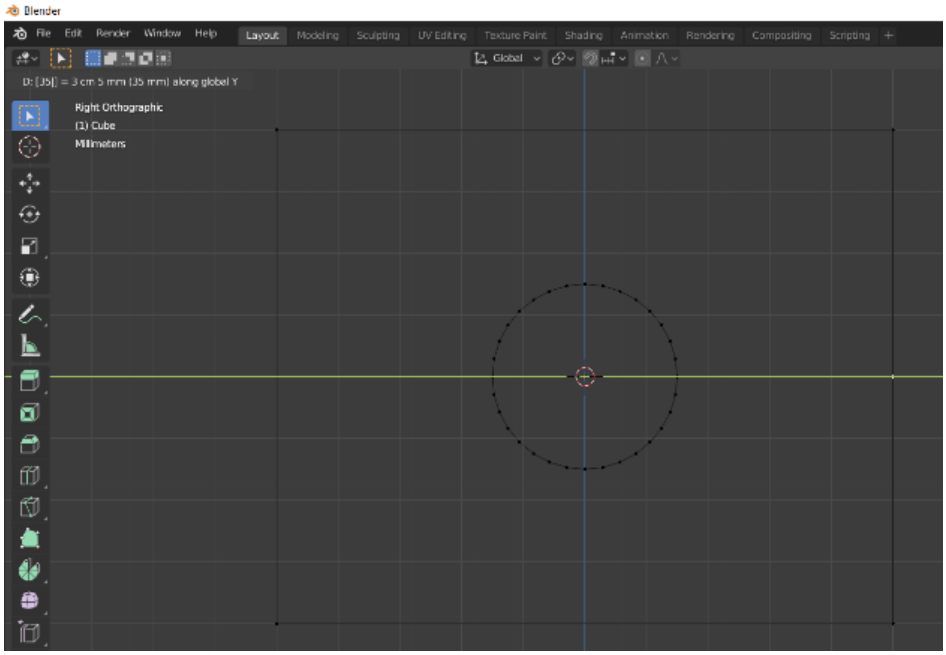


Figure 32. Auxiliary edge for planes

9. The same step for opposite side of circle. And for opposite background.

10. Now as we have auxiliary edges, dividing background plane to two part, we can select separately parts as it shown in *Figure 33a*. Then press “F” (Face). The Face is created between selected points (*Figure 33b*)
11. Make the same step to second part of background. Also the same steps for opposite background.
12. Press **TAB** to switch to *Object* mode.
13. The last step is to export to *.stl file: *File>Export>(*.stl)*
14. The view of final shape in Blender program and exported to *.stl file is shown in *Figure 14.*

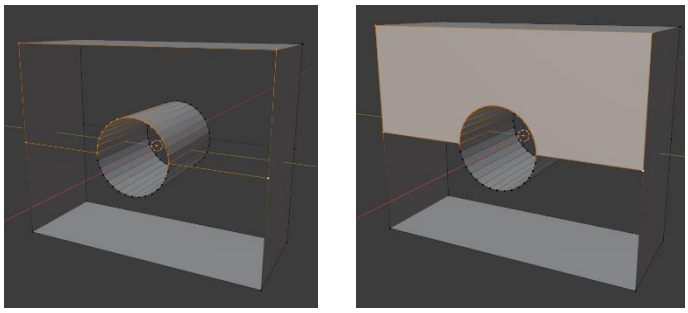


Figure 33. Select points(a) and create Face between them (b)

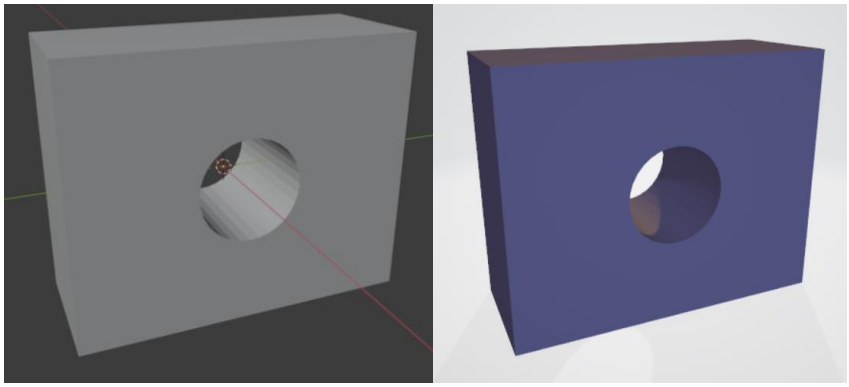


*Figure 34. Final object in Blender enviroment and created *.stl file*

It seems Blender is easer tool to create such model. But you have to know a lot of things to use it, such as different modes for editing, for plane direction and ect.

5. Making the fingers move

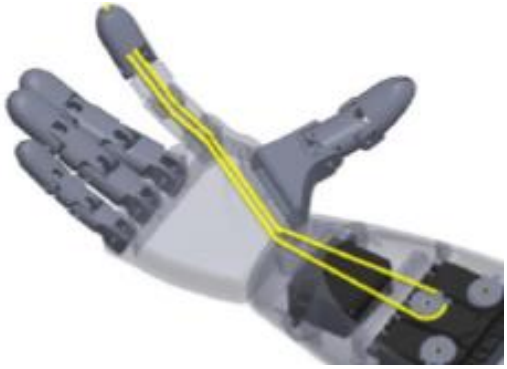


Figure 35. Moving a finger of the bionic hand

Overview

As it was shown before, one of the easiest ways to deploy motions of the bionic hand could be by a servo motors that are pulling strings ("tendons"), connected to furthermost moving part of a finger (see Fig.35).

The motors could be controlled by a comparatively autonomous electronic device- a single board computer (SBC) of a size of a credit card. There are plenty of such devices on the market, but the suggested one is the Arduino UNO microcontroller (see Fig.2), because:

- It is open-source based, comparatively cheap and extremely popular among the enthusiasts in electronics and small automation;
- It is included as one of simulated virtual devices in a Tinker CAD – a web-based free platform for design learning. This is a particular advantage as 3D modelling, circuit and software design could be performed within the same environment. Furthermore, the virtual design, modelling and testing of the hardware and the software will help to avoid save equipment damage (therefore- some expenses), as the virtual hardware elements does not get out of order, if connected and used improperly. Some recommended preventive preparation steps before deploying the design in a real hardware will be mentioned below.

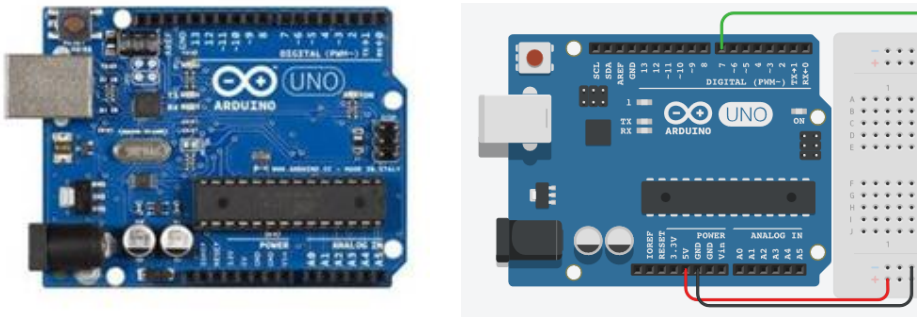


Figure 36. Arduino UNO microcontroller (real view- left, virtualised in TinkerCAD- right)

Arduino UNO microcontroller

The Arduino UNO microcontroller (here and thereafter referenced as "UNO") has limited computational resources comparing to desktop PCs, laptops, tablets or even mobile phones, but it is completely suitable for our task – deploying a controlled motion:

- UNO could be connected to a Windows PC through an USB cable, to supply the power and to load a software program;
- The software is usually developed in a limited version of C programming language inside a special Arduino software development environment called SKETCH. It is worth to mention, that ***the software could also be designed and tested in the Tinker CAD***, then exported as a file, read by SKETCH, compiled and loaded to an UNO device.
- The UNO with loaded program can operate without a PC and control connected devices, when powered independently (7-12 Volts DC power supply or 9V battery recommended).

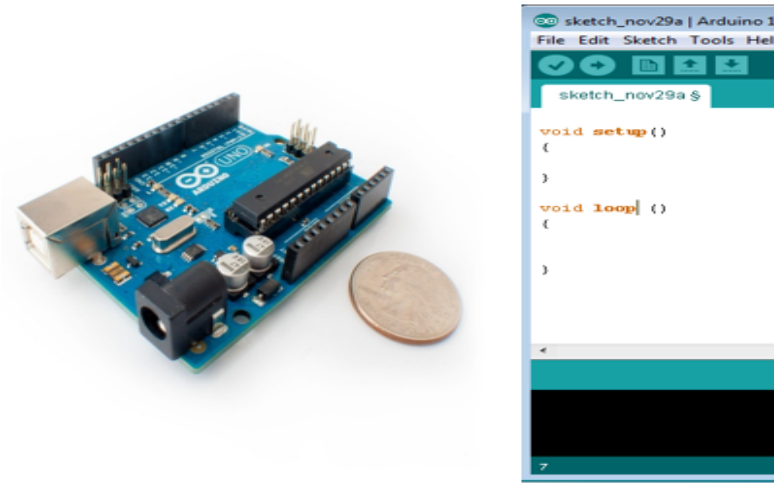


Figure 37. Arduino UNO microcontrolled and SKETCH programe development env procedures executed by UNO, that should be designed: setup() – ini repeated until the power is supplied.

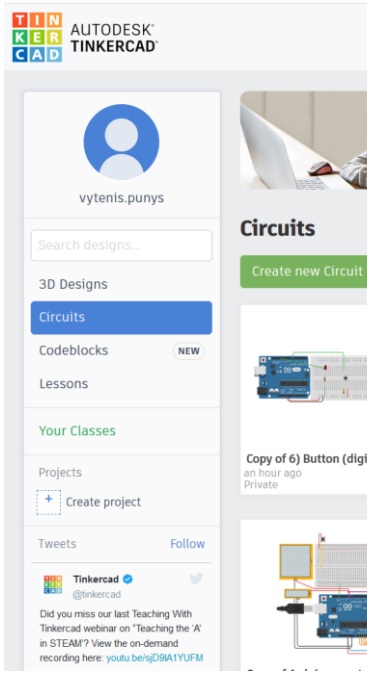


Figure 38. „Circuits" section (left column) in Tinker CAD

Arduino UNO in Tinker CAD

The electronics and software design modelling happens entering the "Circuits" section (Fig. 38) in the Tinker CAD (the "3D design" section might have been used before for modelling parts of fingers). One can find a comprehensive library of Tinker CAD examples and designs for learning.

The graphical development environment of the Tinker Cad contains 2 parts: a virtual electronics on the left and electronic elements collections or programming editor on the right (as seen on Fig. 39). The latter could be used in two alternative modes:

- C program editor, like in SKETCH – one can export a C program file to be loaded to a real UNO;
- Graphical programming environment "CodeBlocks", extremely similar to SCRATCH[2]. The CodeBlocks program is automatically converted to a C program (but it is not a case in opposite direction, one should be careful not to lose own work).

[2] Graphical programming environment, designed by MIT.EDU and widely used to teach programming at schools https://scratch.mit.edu/

There will be 5 servo-motors used to move fingers of the bionic hand in the virtual experiment, schematically shown on Figure 39 and Figure 40. The motors will be controlled and driven by the UNO, and 10 buttons (5 fingers x 2 directions = 2 rows of buttons) will be used to initiate the motion.

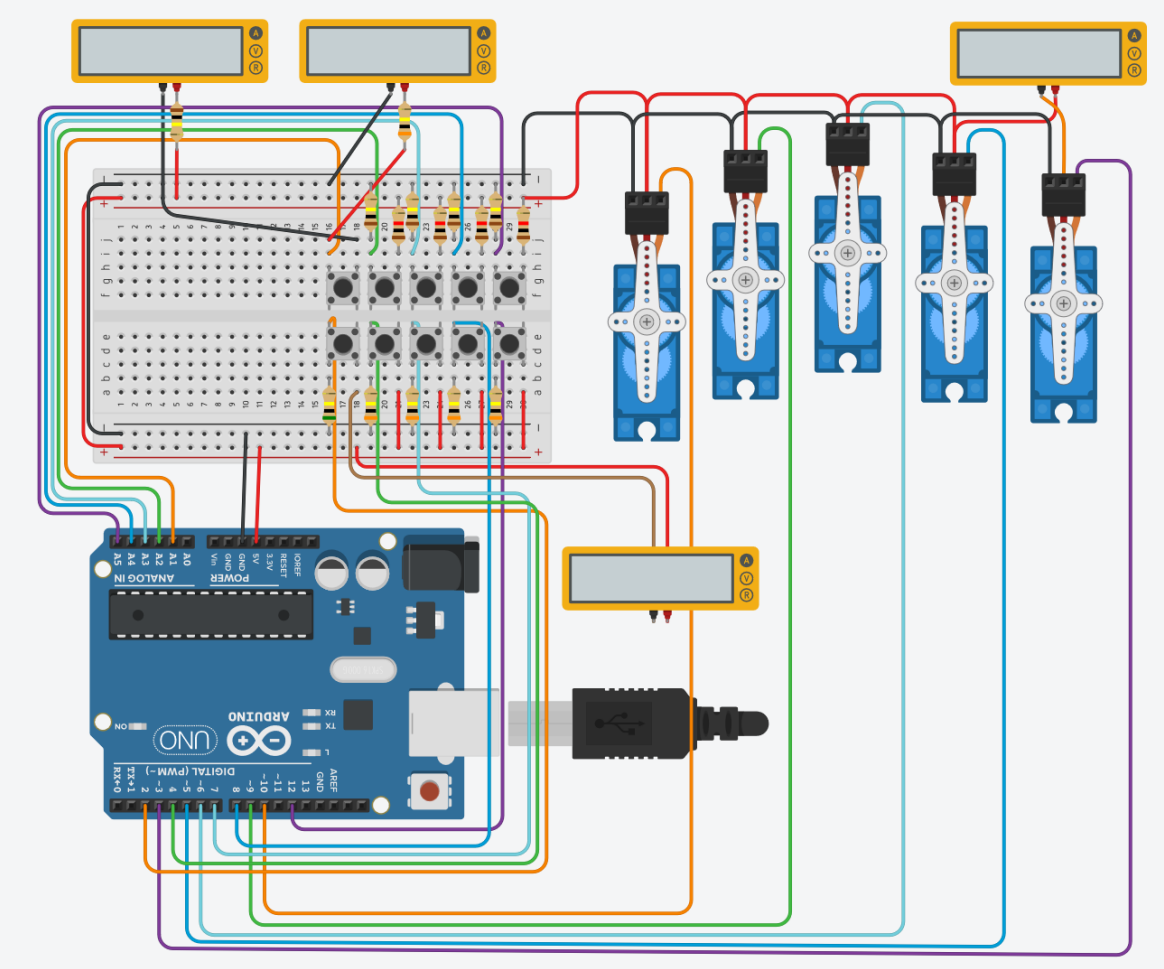


Figure 40. The virtual experiment – moving 5 servo-motors (fingers) by UNO. There are additional multimeters used to check the current through buttons and a servo-motor, in order to avoid possible overload)

The servo-motor is a digital device, turning a spindle to a given position (usually from 0 to 180 degrees). The motors (control pins) should be connected to PWM (Pulse-Width Modulation – the way to specify a rotation position) marked digital outputs of the UNO. Totally, there are 6 such (PWM) digital outputs in an UNO.

One set of buttons (e.g. “Up”) will be connected to analog inputs of an UNO, another set (e.g. “Down”) of buttons - to digital inputs of an UNO. It should be mentioned, that connecting a button to an analog input and reading the value requires slightly more efforts than a to digital one.

Developing a program

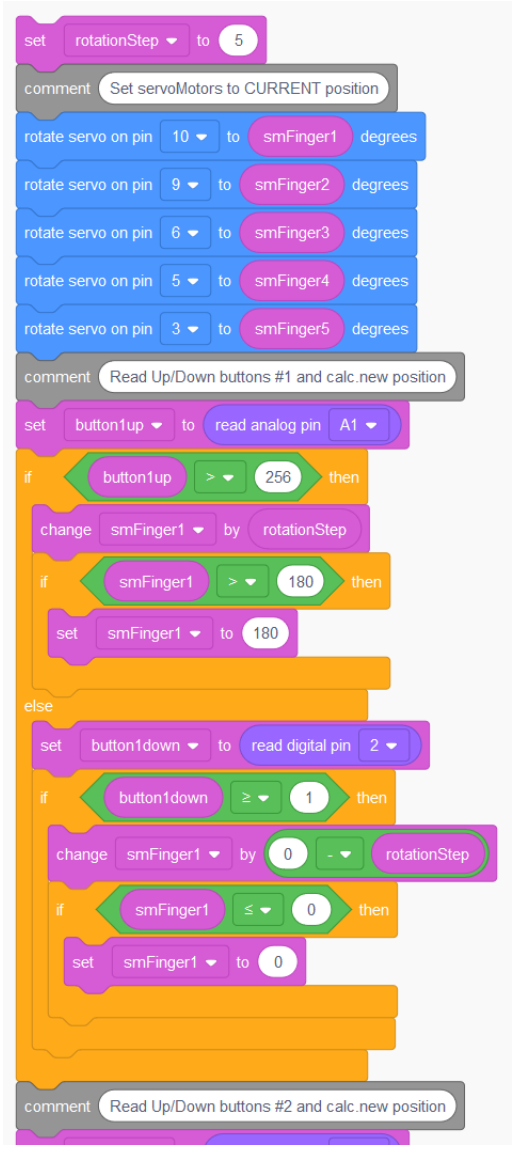
The Tinker CAD graphical programing environment (GPE) is limited to the development of the ***loop()*** function only, the ***setup()*** is deployed as default. That minor limitation could be overcome

continuing further development only in C (no CodeBlocks), what might be acceptable for bigger practical applications. The GPE contains 6 kinds of “CodeBlocks” operations, coded in different colors (Fig. 41a): variables, control, output, input, math (includes calculation of meeting a condition, so called Boolean operations, used within control statements), notation (comments). The statements of different kinds are combined with each other according their shapes.

The program that moves servo-motors (“fingers”) consists of one minor part and two major ones (Fig. 41b), separated by comments in grey:

1) Setting the rotation speed (**rotationStep** variable, pink) – while transferring the program to a real UNO, one can shift this statement into the **setup()** procedure.
2) Giving instructions for each servo-motor to rotate to a certain position – 5 outputs (blue), where one should be careful to specify the correct output pin (on the UNO board) and position (different variables, pink). As these variables are set by default to 0 during the **setup()**, nothing happens after a power-on, until any button is pressed.
3) Checking the button pairs and recalculating the position of servo-motors. This part is repeated (only one visible in Fig. 41b) 5 times for each set of 2 buttons and a servo-motor.
   a) The state (voltage) of button ”Up” is read from an analog input. The possible read-out range is from 0 to 1023 units, corresponding to 0 and 5 Volts respectively. One should be careful to specify the correct input pin on the UNO board.
   b) If the read value is greater than 256 (1.25V), then the position of a particular servo-motor should be recalculated, increasing it by a value of **rotationStep**;
   c) The calculated position should not exceed 180°. If it happened, it is set to this limiting value.
   d) If the button “Up” was not pressed, the **else** branch is executed.
   e) The state of button ”Down” is read from a digital input (0 = No, or 1 = Yes). One should be careful to specify the correct input pin on the UNO board.
   f) If the read value is greater or equal 1, then the position of a particular servo-motor should be recalculated, decreasing it by a value of **rotationStep**. An additional arithmetic (math, green category) operation was necessary to obtain a negative value of **rotationStep**).
   g) The calculated position should not drop below 0°. If it happened, it is set to this limiting value.

(a)



. . .

(b)

Figure 41. Part of graphical program for actuating servo-motors

Proceeding further

There are some precautions one should take into account when trying to assemble a real working hardware model:

- The servo motors are of different power and might require different power supply schemes, as the UNO board may not supply enough power, especially when powered through USB connection.
- In real life it might be useful to employ a force sensor at the end of each finger and use it for stopping a motion. Such sensors could be read through analog inputs, therefore different setup for buttons (rearrangement and additional components) will be necessary.

To begin on your own…

There is a design one is suggested to start with (in the Tinker CAD) presented on Fig. 42:

- Simply take a virtual UNO and a virtual servo-motor.
- Connect power supply and ground from the UNO board to corresponding pins of a servo-motor, and PWM marked output of the UNO to the “Signal” pin of a servo-motor.
- Create a program rotating the servo motor to some position and back.

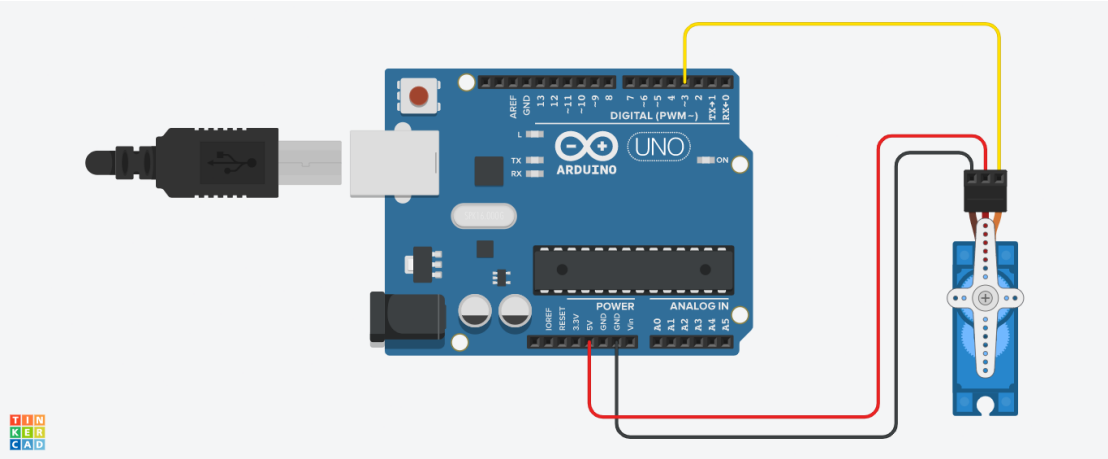


Figure 42. Suggested scheme for the first simulation

Suggested reading

https://dronebotworkshop.com/servo-motors-with-arduino/

Karlsruher Institut für Technologie

kauno technologijos universitetas

CEIPES

UNIVERSITÀ DEGLI STUDI DI PALERMO

Co-funded by the Erasmus+ Programme of the European Union

This project has been funded with support from the European Commission - Application number 2019-1-DE03-KA201-060125.