Erasmus+ program Cooperation partnerships in school education
“Diversifying the STEM Ecosystem”
No 2024-1-LV01-KA220-SCH-000250755

Methodology for Additional Lessons in “Modern Technology Studio” for Ozolnieki Secondary School

Publication was funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the State Education Development Agency (SEDA). Neither the European Union nor SEDA can be held responsible for them.

Content

Introduction

This methodology has been developed within the Erasmus+ programme Cooperation Partnerships in School Education project “Diversifying the STEM Ecosystem” (No. 2024-1-LV01-KA220-SCH-000250755). The program has been created in cooperation between the Ozolnieki Secondary School and academic staff from the Latvia University of Life Sciences and Technologies (LBTU). The authors of the methodology are Natalija Sergejeva, Baiba Briede, Tatjana Rubina, Natalja Vronska, and Jekaterina Smirnova.

The methodology “Modern Technology Studio aims to support students’ digital creativity and technical literacy through modern browser-based tools for image editing, 3D modelling, and multimedia creation. The program combines practical, hands-on activities with digital design thinking to help learners build competence in visual communication, programming logic, and innovation. In addition, the methodology introduces students to the basics of robotics, programming, and control systems.

The methodology combines accessible online tools and physical computing to encourage students’ creativity, problem-solving, and technological fluency. Modern browser-based resources such as iPiccy, Pixlr, and Ezgif introduce students to the fundamentals of image editing, design composition, and animation. Learners explore how to enhance photos, create visual effects, design collages, and produce GIF animations, gaining essential digital literacy skills that bridge art and technology.

Further, SketchUp introduces students to 3D modelling and spatial design. Using intuitive tools for geometry and visualization, students learn to design and print 3D objects such as keychains, vases, boxes, or small houses. This builds spatial reasoning, planning, and precision- skills vital for engineering and architecture.

The course also integrates video creation with Magisto, an AI-powered online editor that transforms photos and clips into polished videos. Through this, students understand narrative flow, visual storytelling, and the growing role of artificial intelligence in digital media.

Beyond creative digital media, the methodology expands into robotics and programming using Python and Arduino. Students learn coding fundamentals, understand variables and loops, and explore how software controls physical components. Working with the Arduino CTC 101 system, learners identify sensors, LEDs, speakers, motors, and control boards to create interactive prototypes and simple robots. Modules progress from introductory experiments with digital signals to project-based learning where students design small robotic mechanisms and interactive electronic games.

By combining digital creativity with robotics and coding, this methodology offers a holistic, interdisciplinary approach to STEM education. Students gain not only technical competence but also confidence in problem-solving, innovation, and teamwork- key abilities for participation in the modern technological ecosystem.

Module 1. Various image editing tools in your browser (10 hours)

Aim: Develop skills in image editing, retouching and enhancement.

Outline:

Module 1.1. **Image editing with *iPiccy***
Module 1.2. **Image editing with *Pixlr***

Time/hours: 10 hours in total.

4 hours for Module 1.1.

6 hours for Module 1.2.

Module 1.1. Image editing with *iPiccy*

Learning Objective:
By the end of the Module 1.1, students will be able to understand how to edit and retouch with the online resource *iPiccy*.

Materials:

- Worksheet "Image editing with *iPiccy*".
- Digital images.
- Devices with internet access (laptops/tablets) for using online resource *iPiccy*[1].

Teaching methods:

- demonstration;
- problem solving;
- practical task;
- learning discussing.

Sources: https://ipiccy.com/

[1] https://ipiccy.com/

Lesson Outline:

1st practical lesson

Initiation phase (10 min.)

- Focusing attention: the teacher shows photos with different quality, brightness, and contrast using a projector.
- About the goal and results to be achieved: the teacher encourages students to formulate the goal of the lesson.

Engagement phase (30 min.)

- Learning new information: The teacher introduces students to the online resource *iPiccy* and *Basic* working tools. Students independently explore the online resource *iPiccy*.
- Compiling information and discussing it with students.

2nd practical lesson

Engagement phase (25 min.)

- Use of new information: students work individually with worksheet "Image editing with *iPiccy*" tasks 1-2.

Reflection phase (15 min.)

- Feedback: What did you learn about online resource *iPiccy* today? What worked? What didn't work? What else do you plan to learn?

3rd practical lesson

Engagement phase (40 min.)

- Updating previous knowledge about various *iPiccy's* tools.
- Discussing with students about brightness and contrast. How can it be improved? Why is it necessary?
- Learning new information: the teacher introduces students to the online resource *iPiccy's Photo Effects* working tools. Use of new information: students work individually with task "Image editing with *iPiccy*" tasks 3-4.

4th practical lesson

Engagement phase (25 min.)

- Learning new information: the teacher introduces students to the online resource *iPiccy's Retouch* working tools. The teacher introduces students to video creating of girl's picture using artificial intelligence. Use of new information: students work individually with task "Image editing with *iPiccy*" tasks 5-6.

Reflection phase (15 min.)

- Feedback: What worked? What didn't work? Would you like to use this online resource in the future? Why or why not?

Worksheet "Image editing with iPiccy*"*

Task 1. The photo needs different corrections

For example

beginning

→

finish

- Need to straighten the photo (tool *Rotate & Flip*), changing the *Straighten*: 3, then press *Apply*.
- Freely (as you like) crop unnecessary edges of the photo (tool *Crop Picture*), then press *Apply*.
- Need to change the brightness and contrast of the photo (tool *Light & Contrast*), changing the *Brightness*: 39 and *Contrast*: 4, then press *Apply*.
- Need to improve colour intensity (tool *Colors*), changing the *Temperature*: 34.
- Need to blur the background (tool *Blur Image*), changing the Strength: 60, then choose a brush (*Effect mask/ Original*), and paint the glass, restoring the visibility of the glass. If you need to blur the background fragment again, you need to change the brush: *Effect mask/ Effect.*
- Need to create a black and white accent on the selected (as you like) object (tool *Black & White*), then choose a brush and paint the glass, restoring the colour of the glass.

Tasks 2. The photo needs to have the unnecessary object removed (any image)

For example

beginning

→

finish

- Find and download any image from the free stock image discovery platform, e.g., http://www.unsplash.com or http://www.pixabay.com.
- Need to choose a tool *Clone*, changing the *Brush* size, then click with the green circle where is located the unnecessary object, and then click with the white circle where the cloning area will be.

Task 3. Hiding important information in a photo (any auto)

For example

beginning

finish

- Find and download any image from the free stock image discovery platform, e.g., http://www.unsplash.com or http://www.pixabay.com.
- On the left side, switch the tools to *Photo Effects*
- Need to put pixels, e.g., on a BMW number (tool *Focal Pixelate*), changing the *Focal* size: 10, *Focal hardness*: 50, *Fade*: 0, *Reverse effect*. Repeat this 3-4 times.

Task 4. Creating a collage (any 4 images)

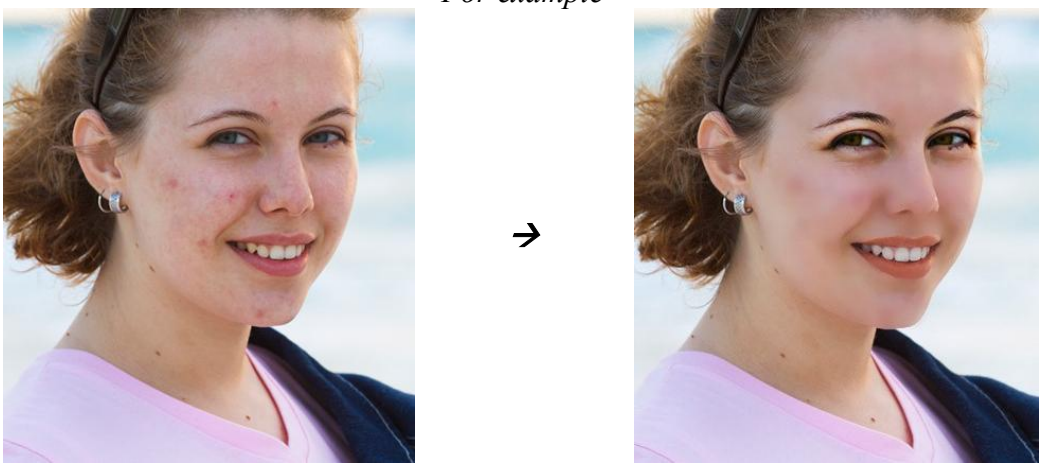
For example

- Find and download any 4 images from the free stock image discovery platform, e.g., http://www.unsplash.com or http://www.pixabay.com.
- On the left side, switch the tools to *Photo Effects.*
- For first image: need to use tool *Orton Effect*, changing the *Bloom*: 50, *Brightness*: 80, *Fade*: 0. Save to computer.
- For second image: need to use tool *Retro Comic*, changing the *Dot* size: 5, *Angle*: 35, *Brightness*, *Contrast*, *Fade*: 0, *Blend Mode*: *Color*. Save to computer.
- For third image: need to use tool Rainbow, changing the mainot *Fade*: 0, *Blend Mode*: Multiply. And tool *Scanlines*, mainot iestatījumus *Line* size: 11, *Fade*: 50. Save to computer.

- For fourth image: need to use tool *Duotone*, changing the First: blue, Second: green, Contrast: 20, Fade: 0. And tool Gritty, changing the Darkness: 50, Noise: 70, Fade: 0. Save to computer.
- Switch to the collage maker (*Editor/ Collage*), and choose a collage layout, e.g., 4 images (1 image is at the top, 3 images are at the bottom). Add the newly enhanced images. Can create a horizontal mirror of any of the images.
- In the collage settings, change the *Spacing*: 20, *Roundness*: 50. Change background colour to the one you like best.

Task 5. Photo retouching

For example



beginning *finish*

- On the left side, switch the tools to *Retouch*
- Need to remove pumps (tool *Blemish Fixer*).
- Need to smooth out face skin (tool *Airbrush*), changing the *Brush* size: 10 and *Fade*: 20%. Teeth, eyes, eyebrows – don't touch!
- Need to create more expressive eyes (tool *Mascara*), changing the *Brush* size: 7 and *Fade*: 35%.
- Change eye colour to the one you like best (tool *EyeColor*), changing the *Brush* size: 10 and *Fade*: 20%.
- Need to whiten teeth (tool *Teeth Whiten*), changing the *Brush* size: 7 un *Fade*: 0 %.
- Paint your lips, changing to the color you like best (tool *Lip Color*), changing the *Brush* size: 4 un *Fade*: 40 %.

Task 6. Create a video of this picture with AI

- Create a video of this picture using artificial intelligence (www.vidu.com) (need *Google* or *Apple* account).

- Choose *Image to Video*, then add the girls's photo in JPG extension, change *Model* to *Vidu 1.5* and press creating button.
- Continue, choose *Reference to Video*, then add the girl's photo in JPG extension, write additional details below the photo, e.g., the girl has to turn to camera, smile, and put on sunglasses, then press creating button.

Module 1.2. Image editing with *Pixlr Editor*

Learning Objective:

By the end of the Module 1.2, students will be able to understand how to edit and retouch with the online resource *Pixlr Editor*.

Materials:

- Worksheet "Image editing with *Pixlr Editor*".
- Digital images.
- Devices with internet access (laptops/tablets) for using online resource *Pixlr Editor*[2].

Teaching methods:

- demonstration;
- problem solving;
- practical task;
- learning discussing.

Sources: https://pixlr.com/editor/

Lesson Outline:

1st practical lesson

Initiation phase (10 min.)

- Focusing attention: the teacher shows photos with different quality, complexity, and contrast using a projector.
- About the goal and results to be achieved: the teacher encourages students to formulate the goal of the module.

Engagement phase (30 min.)

- Learning new information: the teacher introduces students to the online resource *Pixlr* with working tools and many panels: *Layers*, *History*, *Adjustment* and *Filter*. Students independently explore the online resource *Pixlr*. Compiling information and discussing it with students.

2nd practical lesson

Engagement phase (25 min.)

[2] https://pixlr.com/

- Use of new information: students work individually with worksheet “Image editing with *Pixlr*” tasks 1-2.

Reflection phase (15 min.)

- Feedback: What did you learn about online resource *Pixlr* today? What worked? What didn't work? What else do you plan to learn?

3rd practical lesson

Engagement phase (40 min.)

- Updating previous knowledge about various *Pixlr's* tools. Discussion with students about the similarity of the *Pixlr* online resource to *Photoshop*. About the *Layers* panel and the need to always see it. How can it be added if the *Layer* panel is not by default? Why is it necessary? Where can be changed brightness, contrast, colors balance, hue and saturation?
- Learning new information: the teacher introduces students to the online resource *Pixlr's* blend mode options and opacity.
- Use of new information: students work individually with task “Image editing with *Pixlr*” tasks 3-4.

4th practical lesson

Engagement phase (30 min.)

- Use of new information: students finished individually with task “Image editing with *iPiccy*” task 4.

Reflection phase (10 min.)

- Feedback: What worked? What didn't work?

5th practical lesson

Engagement phase (40 min.)

- Updating previous knowledge about various *Pixlr's* tools. Discussion with students about Filters. Where can find the blend mode and opacity? How can I change the name of a layer?
- Learning new information: the teacher introduces students to a free online GIF maker. The teacher demonstrates the created GIF animation.
- Use of new information: students work individually on the task 5.

6th practical lesson

Engagement phase (30 min.)

- Use of new information: students individually complete task 5.

Reflection phase (10 min.)

- Feedback: What worked? What didn't work? Would you like to use this online resource in the future? Why or why not?

Worksheet “Image editing with Pixlr”

Task 1. Convert a colour image to black and white, with the colour effect (any image)

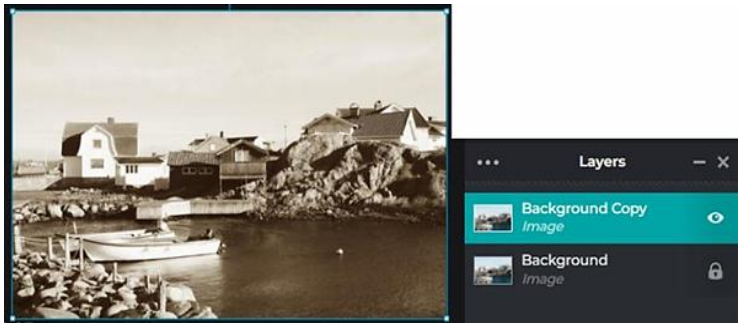
e.g. 

- Find and download any 4 images from the free stock image discovery platform, e.g., http://www.unsplash.com or http://www.pixabay.com.
- Open *Pixlr editor,* then choose *Open Image* to open this image. In the *Layers* panel unlock the background layer (click on the key) and duplicate it (*Cmd+D* or *Ctrl+D*).
- Correction of the duplicate layer (*Adjustment/ Desaturate)* and use the eraser (*Softness 0%*) to return colour to more objects of your choice.
- The coloured layer needs to change colour saturation: *Adjustment/ Hue and saturation* changing the *Saturation*: 25-30.

Task 2. Creating an old photo (any image)

e.g. 

- Find and download any 4 images from the free stock image discovery platform, e.g., http://www.unsplash.com or http://www.pixabay.com.
- Open *Pixlr editor*, then choose *Open Image* to open this image. In the *Layers* panel unlock the background layer (click on the key) and duplicate it (*Cmd+D* or *Ctrl+D*).
- To make the image more yellow, you need to make the following changes: *Adjustments/ Hue and saturation*, need ticking *Colorize*, change *Hue*: -140, *Saturation*: -50, *Lightness*: 0:

e.g. 

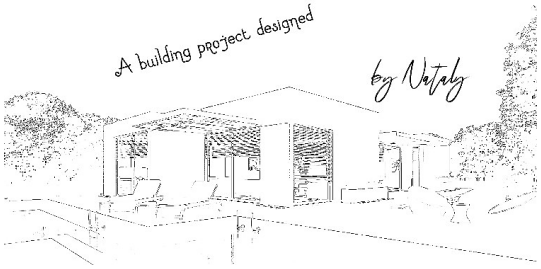
- Apply the filter *Effect library/ Retro/ Ragwarm* to the *Background Copy* layer. Then apply the filter *Details/ Add Noise* changing the *Amount*: 20.
- Using the tool *Marquee Select*, select the left side of the image and press *Delete*. To stop the marquee from blinking, press *Cmd+D* or *Ctrl+D*.

Task 3. Sketch from a photo (any photo with good black&white contrast)

For example

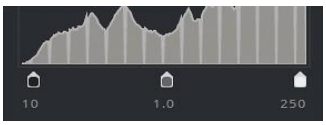
beginning

→



finish

- Find and download any image from the free stock image discovery platform, e.g., http://www.unsplash.com or http://www.pixabay.com.
- Duplicate the *Background* layer, convert the colour image to black and white (*Adjustments/Desaturate*).
- Run *Adjustments/Levels* and change the following settings:

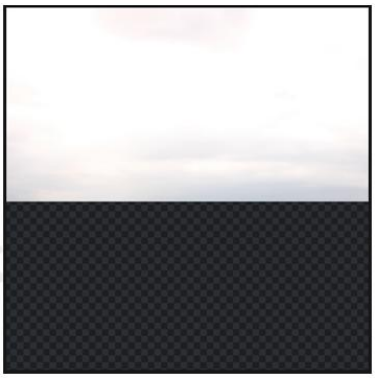


- Duplicate the *Background* layer and move this duplicate above all existing layers.
- Apply the *Find Edges* filter, then invert *Adjustments/Invert*.
- Apply the *Effect library/ Urban/ Sham* filter.
- Need to improve colour saturation, contrast and lightness *Adjustments/Hue & Saturation*, changing the *Hue*: 0, *Saturation*: 0, *Lightness*: +15; then *Brightness & Contrast,* changing the *Brightness*: 0, *Contrast:* +50.
- Change the blend mode to *Luminosity*.

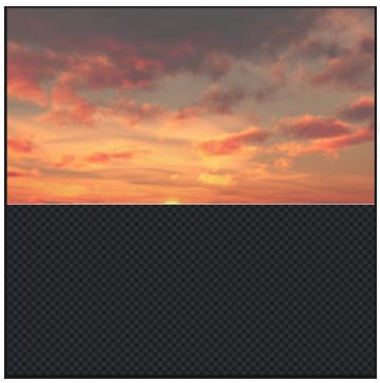
Task 4. Creating a life book

e.g. 

- Open a new file with the following parameters *Width*: 3000 px, *Height*: 3000 px, *Background*: none.
- Open the image *D&Z* (*Full HD*), select only the sky. After selecting, copy the selected fragment and move it to the main file (*Untitled*), change the dimensions. Rename the layer to *Sky*.

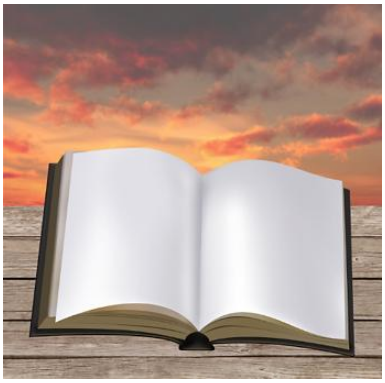


- Open the image *Sunset*, select the upper part of the sunset. After selecting, copy the selected fragment and move it to the main file (*Untitled*), change the dimensions. Change the name of the layer to *Sunset*. Change the *opacity* to 65%.

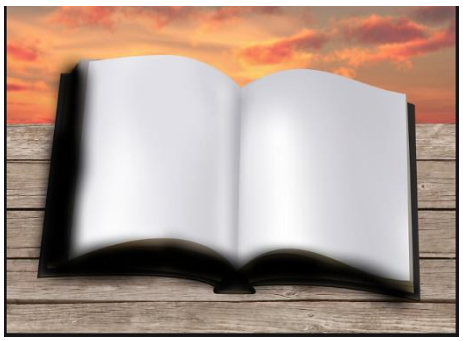


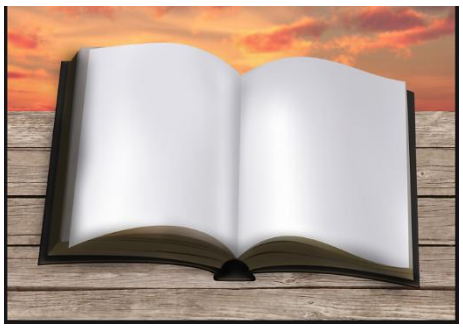
- Open the image *Table*, select everything and move it to the main file (*Untitled*), changing the dimensions if necessary. Rename the layer to *Table*.

- Open the image *Book* (*Full HD*), select everything and move it to the main file (*Untitled*), change the dimensions and rotate if necessary. Change the name of the layer to *Book*.



- The book needs to have a shadow. Create a new layer (*Layer/New layer*) and rename it *Shadow*. Select the brush and set the following parameters the Size: 400 px, Softness: 100%, Step: 10%. Change the *Opacity* to 65%.





Painted with a brush *Applied opacity*

- Open the image *Road* (*Full HD*) and use the *Lasso* tool to select only the road. After selecting, copy the selected fragment and move it to the main file (Untitled), changing the dimensions if necessary. Change the name of the layer to *Road*.
- Select the eraser tool (*Softness*: 76%, *Step*: 0%) and erase the edges of the road around the perimeter to make it look more natural.

- Open the image *Fox*, use the *Select/Subject* to select only the fox. After selecting, copy the selected fragment and move it to the main file (Untitled), changing the dimensions if necessary. Change the name of the layer to *Fox*. Write your name on the left side of the book.
- Convert the road to black and white and use the free stock image discovery platform, e.g., http://www.unsplash.com or http://www.pixabay.com to place any object.

Task 5. Creating GIF animation (any several consecutive photos)

- Open a new file. Select the option to write text and write the name of the animation. Using the *Snipping* tool, save it with the name *1-frame*.
- Open the first image (*File/Open Image*), select (*Marquee Select*) and copy this image, paste into the *background* layer. Add explanatory text below this image.

e.g.

- Open the second image and repeat the previous action. Do this with all your photos.
- Open the animation creator https://ezgif.com/
- Select *GIF Maker* and upload all prepared frames. Press the *Upload files!* button. If your frames are not equally sized, you need to agree to automatically crop/resize all frames to match the smallest one.
- Set the frame *Delay time* for the animation: 700. Press the *Make a GIF!* button.
- You can apply *Effects* to a newly created GIF animation.

- Use the created GIF animation, you should create an MP4 video. Press *Video to GIF*, then upload GIF animation and upload video.

Module 2. 3D modelling with 3D printing (18 hours)

Aim: Develop spatial thinking, creativity, and digital skills in content creation. Understand how to create a model in the free online program *SketchUp,* how to create animations from multiple scenes and to print the created model with a 3D printer.

Outline:

Module 2.1. Key ring (keychain) with your name
Module 2.2. Vase creation
Module 2.3. Storage box creation
Module 2.4. 3D house creation with animation

Time/hours: 18 hours in total.

4 hours for Module 2.1.

2 hours for Module 2.2.

4 hours for Module 2.3.

8 hours for Module 2.4.

Module 2.1. Key ring (keychain) with your name

Learning Objective:
By the end of the lesson, students will be able to create a key ring (keychain) model in the online program *SketchUp* and print key ring (keychain) with a 3D printer.

Materials:

- projector;
- devices with internet access (laptops/tablets) for using online resource *SketchUp*[3]
- practical task "Key ring (keychain) with your name";
- 3D printer.

Teaching methods:

- demonstration;
- practical task;
- problem solving;
- learning discussing.

Sources: https://sketchup.trimble.com/en/try-sketchup

1st practical lesson

[3] https://sketchup.trimble.com/en/try-sketchup

Initiation phase (10 min.)

- Focusing attention: the teacher shows various models printed with a 3D printer.
- About the goal and results to be achieved: the teacher encourages students to formulate the goal of the module.

Engagement phase (30 min.)

- Learning new information: the teacher shows students how to connect to the online resource *SketchUp*. Explains how to connect to the free version using an e-mail, *Google, Apple* or *Microsoft* account. The teacher begins introducing students to the online resource *SketchUp* working tools (*Line, Rectangle, Push/Pull, Rotate, Orbit, Pan, Paint, Scale*) and the ribbon *Model Info.* Compiling information and discussing it with students.

2nd practical lesson

Engagement phase (25 min.)

- Learning new information: the teacher shows students how to connect to the online resource *SketchUp*. Explains how to connect to the free version using an e-mail, *Google, Apple* or *Microsoft* account. The teacher continues introducing students to the online resource *SketchUp* working tools (*Tape Measure, Move, 2 Point Arc, Dimensions, 3D text*) and the ribbon *Display.*
- Use of new information: students independently explore the online resource *SketchUp.*

Reflection phase (15 min.)

- Feedback: What did you learn about online resource *SketchUp* today? What worked? What didn't work? What else do you plan to learn?

3rd practical lesson

Engagement phase (40 min.)

- Updating previous knowledge about various *SketchUp* tools and ribbons. Discussing with students about moving tools. How to work with them? How to write measurements for a model correctly? How do *Push/Pull* and *2 Point Arc* tools work?
- Learning new information: the teacher demonstrates to students how to create rounded corners and spatial text. Use of new information: students work individually on the task "Key ring (keychain) with your name".

4th practical lesson

Engagement phase (30 min.)

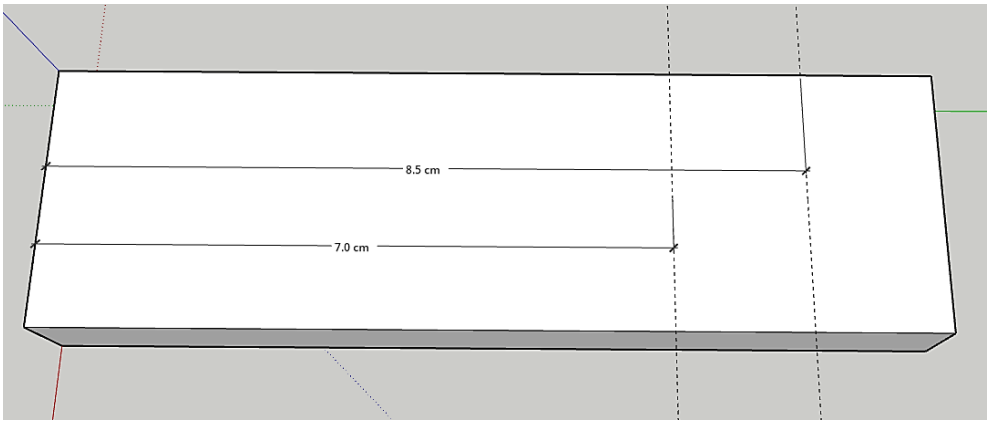
- Use of new information: students continue to work individually on the task "Key ring (keychain) with your name" and print key ring (keychain) with a 3D printer.

Reflection phase (10 min.)

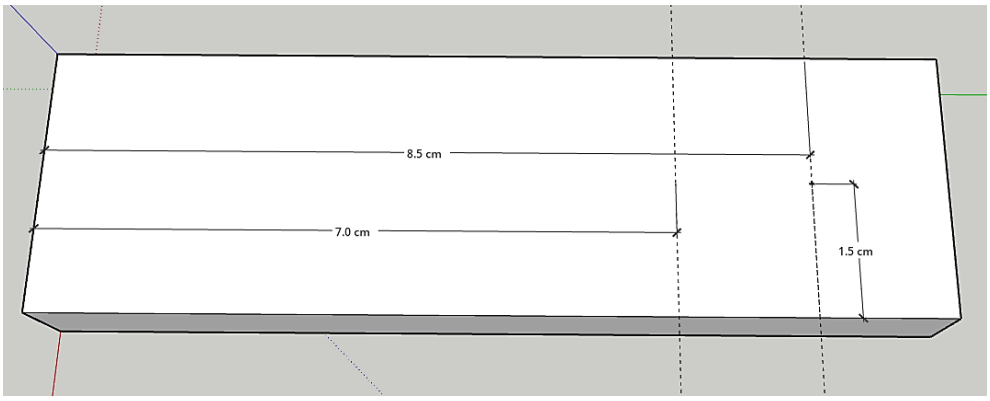
- Feedback: What worked? What didn't work? Was it interesting to create a 3D model and print it? Why or why not?

Worksheet "Key ring (keychain) with your name"

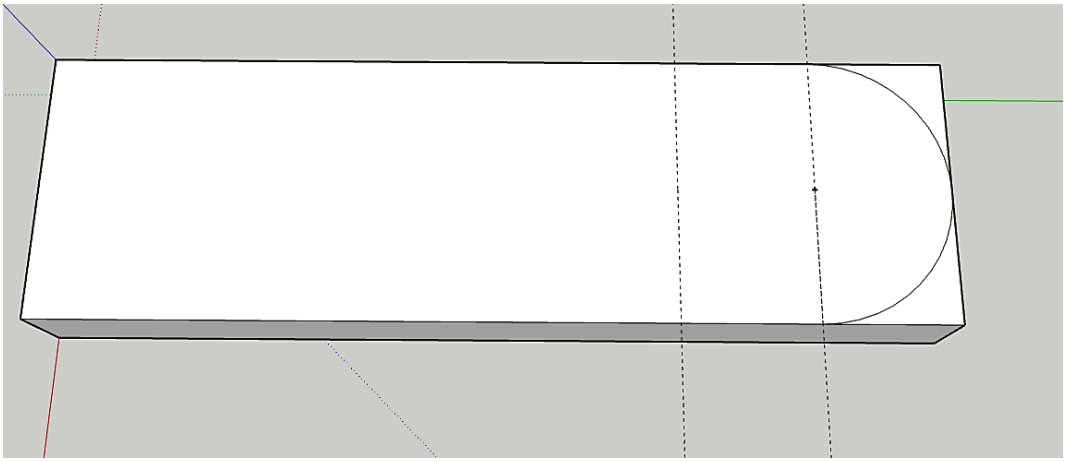
- Open *SketchUp*.
- *Model Info* change *Format*: *Centimeter*, *Display Precision*: 0.0 cm, *Length snapping*: 0.1 cm.
- Select the *Rectangle* tool, click on the starting point and input: 3, 10 cm.
- Select the *Push/Pull* tool and push the model up to a height of 1 cm.
- Select the *Tape Measure* tool, click on the midpoint, move the mouse from the left side to to the center of the model, and insert: 7 cm.
- Select the *Tape Measure* tool again, click in the midpoint, move the mouse from the left side to to the center of the model, and insert 8.5 cm:



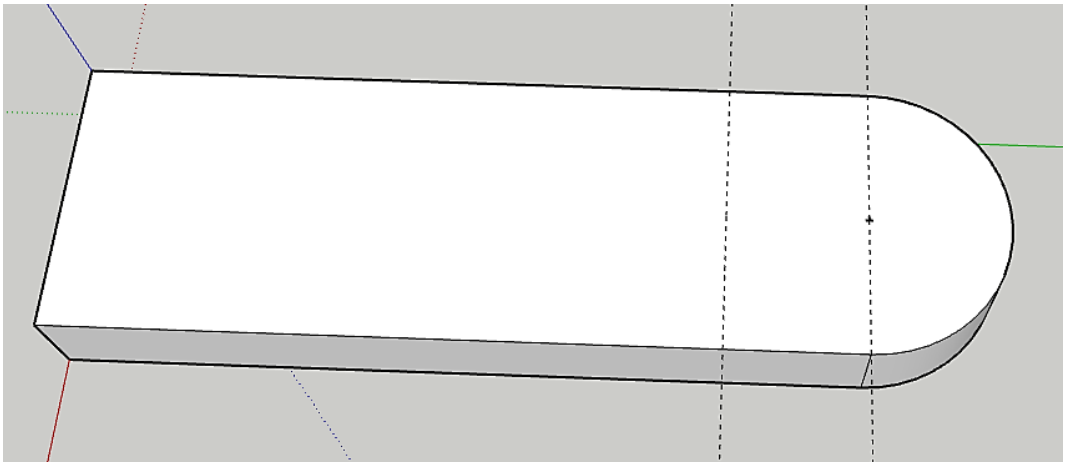
- Select the *Tape Measure* tool again, click and move the mouse from the bottom to the center of the model and insert 1.5 cm:



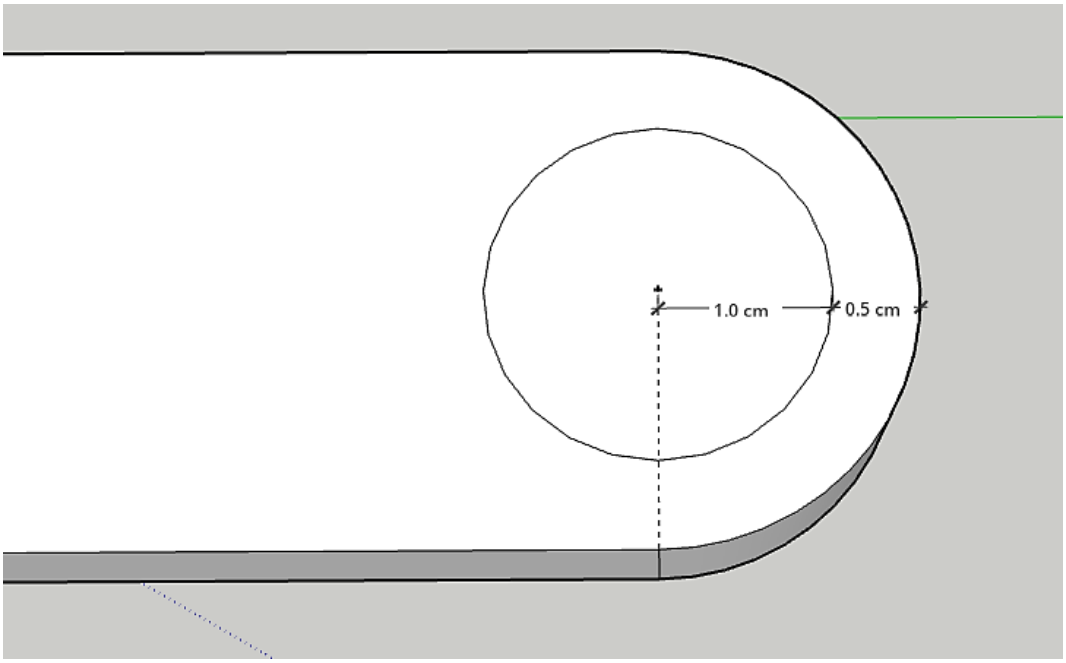
- Select *2 Point Arc* tool, sketch an arc to the midpoint of the model. Then repeat the action, drawing a second arcs:



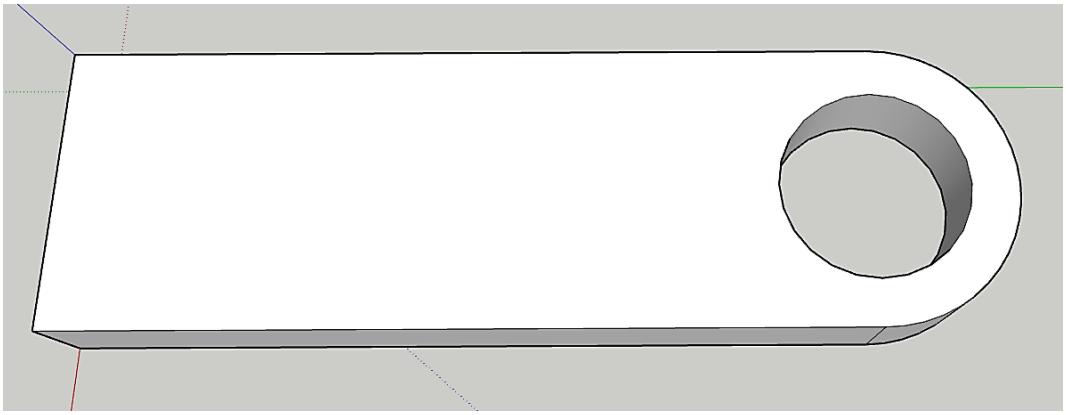
- Push unnecessary material with the *Push/Pull* tool:



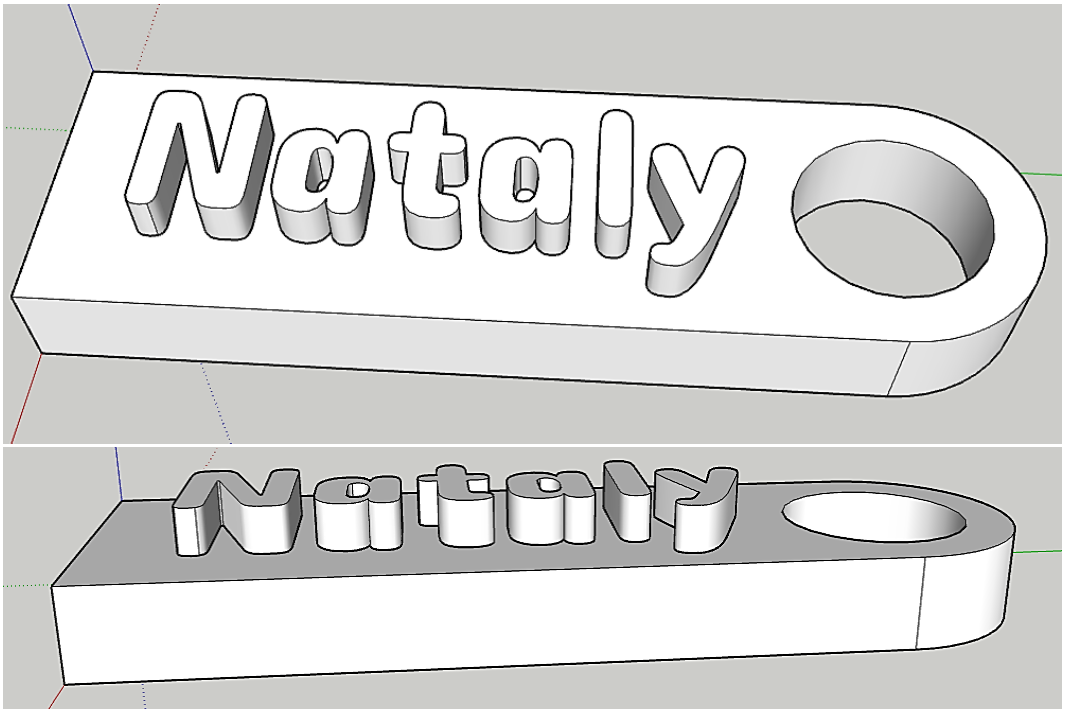
- Using the *Circle* tool and a point on the model, sketch a circle with a radius 1 cm:



- Push the hole with the *Push/Pull* tool:



- Select the *3D text* tool, write your name, and change the parameters, e.g.: font *Concert One*, *Height*: 1.5 cm, *Text extrusion*: 0.5 cm. Then insert your 3D name on the key ring (keychain):



- Save your work as *SketchUp* (*Download/ SKP*).
- Save your work as a 3D model for printing with a 3D printer (*Download/ STL*).
- Print key ring (keychain) with a 3D printer.

Module 2.2. Vase creation

Learning Objective:

By the end of the lesson, students will be able to create a vase model in the online program *SketchUp* and print a vase model with a 3D printer.

Materials:

- projector;
- devices with internet access (laptops/tablets) for using online resource *SketchUp*[4]
- practical task "Vase creation";
- 3D printer.

Teaching methods:

- demonstration;
- practical task;
- problem solving;
- learning discussing.

Sources: https://sketchup.trimble.com/en/try-sketchup

1st practical lesson

Initiation phase (10 min.)

- Updating previous knowledge about various *SketchUp* tools. Discussion with students about moving tools. Where can I find additional tools?

Engagement phase (30 min.)

- Learning new information: the teacher introduces students to the online resource *SketchUp* working tools (*Arc, Circle, Freehand, Offset, Paint*) and the ribbon *Materials*.
- Use of new information: students work individually with task "Vase creation" practical work.

2nd practical lesson

Engagement phase (25 min.)

- Students finished individually "Vase creation" practical work.

Reflection phase (15 min.)

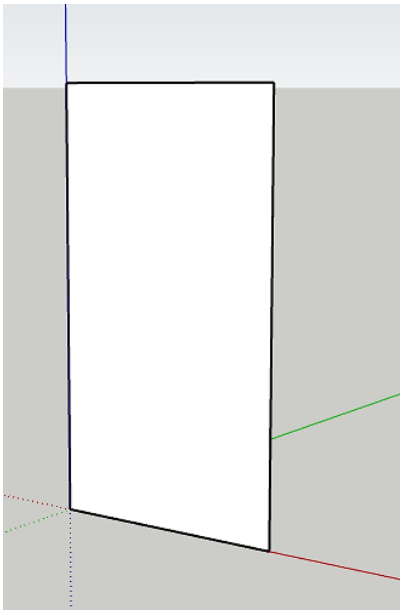
- Feedback: What worked? What didn't work? Would you like to use this online resource in the future? Why or why not?

Worksheet "Vase creation"

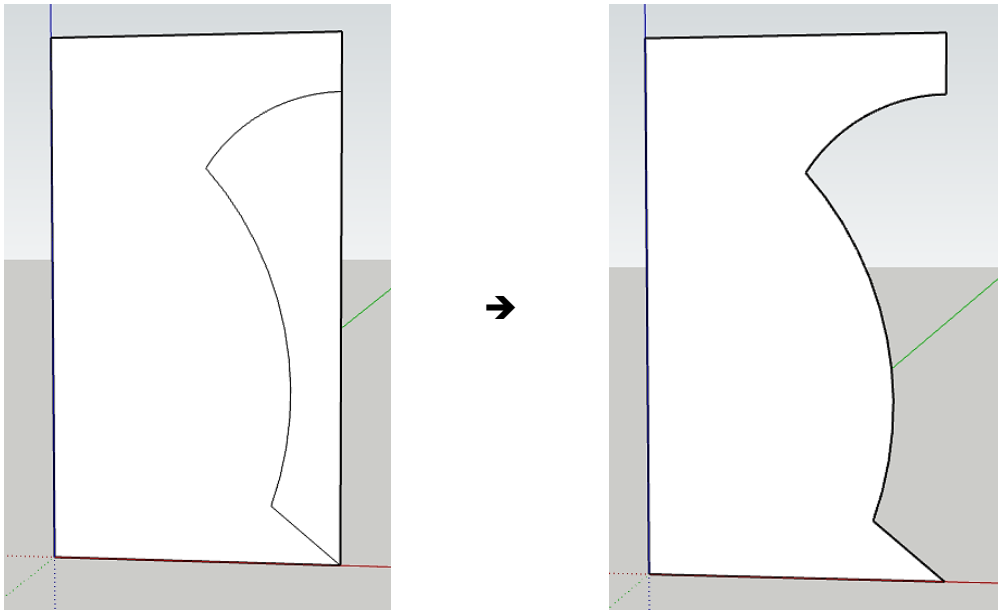
- Open *SketchUp*.

[4] https://sketchup.trimble.com/en/try-sketchup

- *Model Info* change *Format*: *Centimeter*, *Display Precision*: 0.0 cm, *Length snapping*: 0.1 cm.
- Select the *Rectangle* tool, click on the starting point and input, e.g 10, 18 cm.

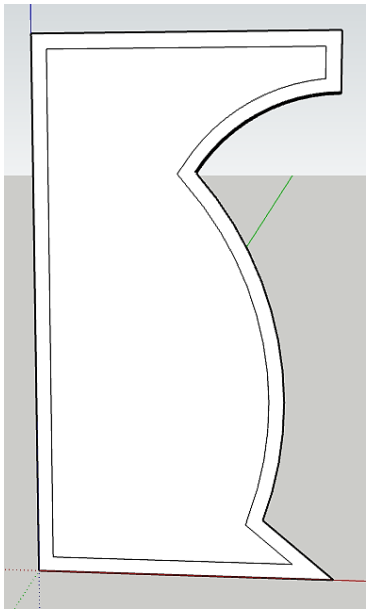


- Using the *Arc* tool and *Line* tool, sketch the profile of a vase (as you wish/free style), then you can delete unnecessary parts of the rectangle, e.g.:

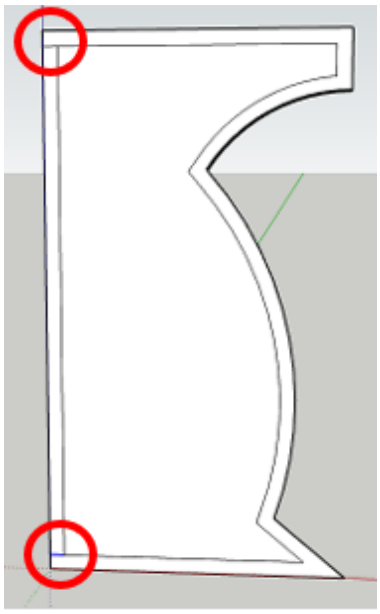


The profile of the vase *Delete unnecessary parts*

- Using the *Offset* tool, the sides of the vase should be thickened: 0.5 cm

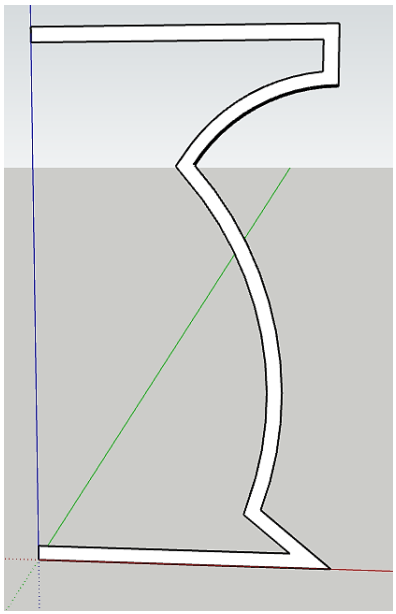


- You should create a contour for the contouring tool, so you need to draw two additional lines. Then you can delete unnecessary parts of the vase.



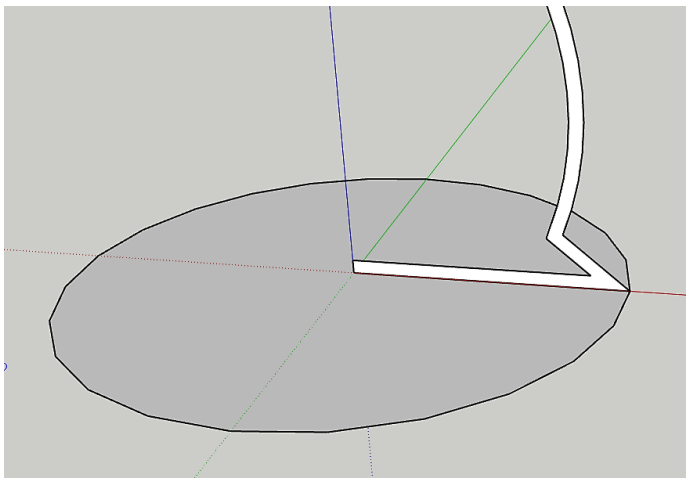
Additional lines

➔

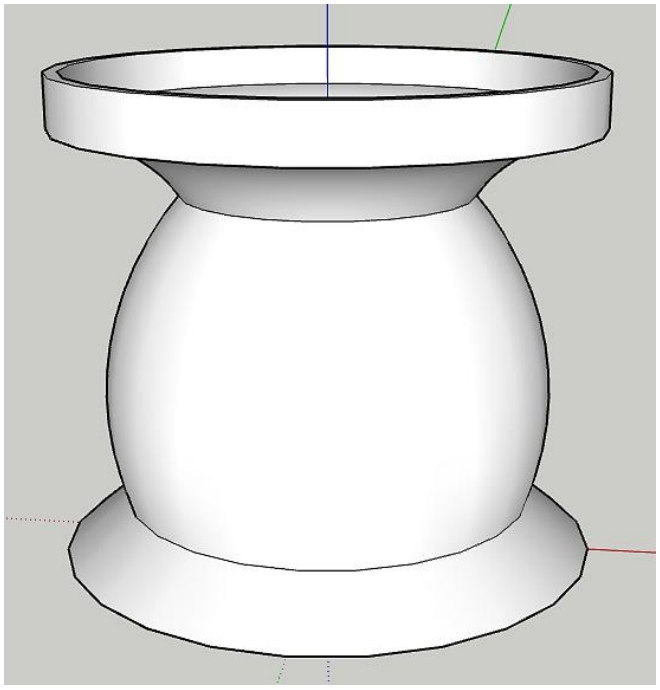


Delete unnecessary parts

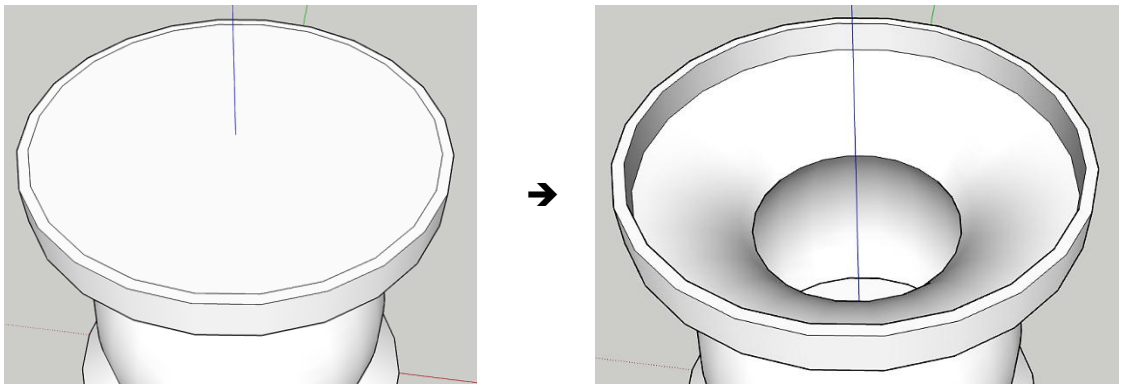
- Use the *Circle* tool to sketch a circle. Make sure that the circle is sketched on the correct plane:



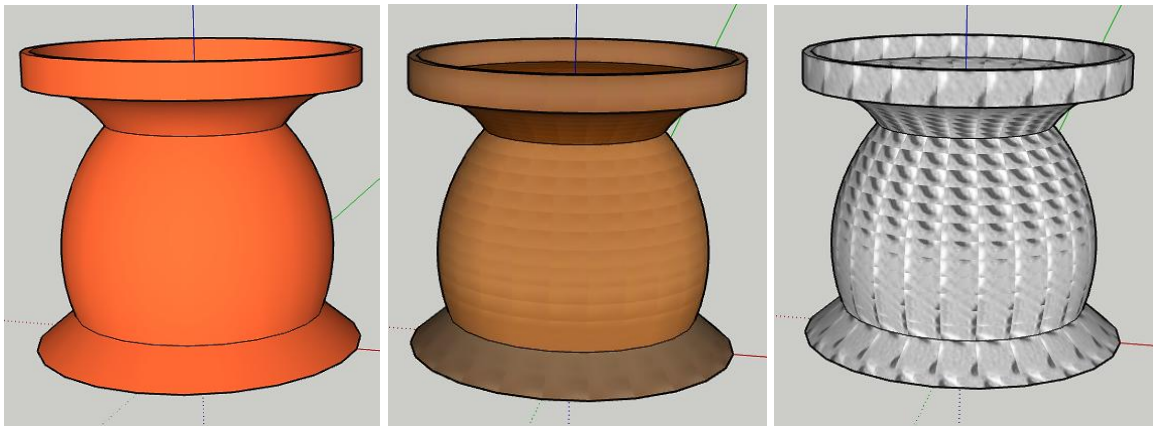
- Create a vase using the *Follow me* tool:



- If there is no hole at the top of the vase, highlight the upper circle and delete it:



- Using ribbon *Materials*/ *Browse*, give the vase any material of your choice, e.g.:



- Save your work as *SketchUp* (*Download/ SKP*).
- Save your work as a 3D model for printing with a 3D printer (*Download/ STL*).
- Print vase with a 3D printer.

Module 2.3. Storage box creation

Learning Objective:
By the end of the lesson, students will be able to create a storage box model in the online program *SketchUp* and print a storage box model with a 3D printer.

Materials:
- projector;
- devices with internet access (laptops/tablets) for using online resource *SketchUp*[5]
- practical task "Storage box creation";
- 3D printer.

Teaching methods:
- demonstration;
- practical task;
- problem solving;
- learning discussing.

Sources: https://sketchup.trimble.com/en/try-sketchup

1st – 2nd practical lesson
Initiation phase (10 min.)
- Updating previous knowledge about various *SketchUp* tools. Discussion with students about moving tools. Where can I find additional tools?

[5] https://sketchup.trimble.com/en/try-sketchup

Engagement phase (30 min.)

- Learning new information: the teacher introduces students to the online resource *SketchUp* working tools (*2 Point Arc, Move*) and the ability to duplicate object.
- Use of new information: students work individually on the task "Storage box creation" as practical work.

3rd –4th practical lesson

Engagement phase (25 min.)

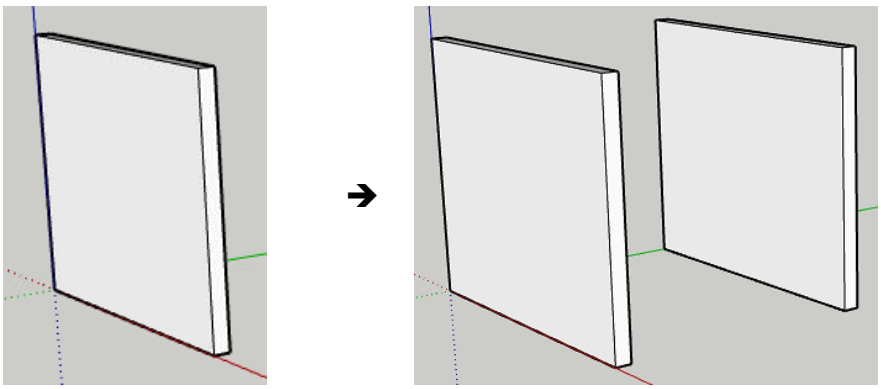
- Use of new information: students finished individually "Storage box creation" practical work.

Reflection phase (15 min.)

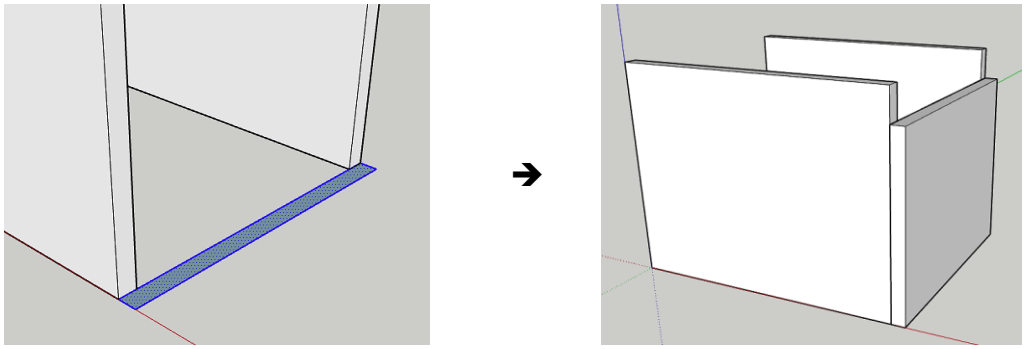
- Feedback: What worked? What didn't work? Would you like to use this online resource in the future? Why or why not?

Worksheet "Storage box creation"

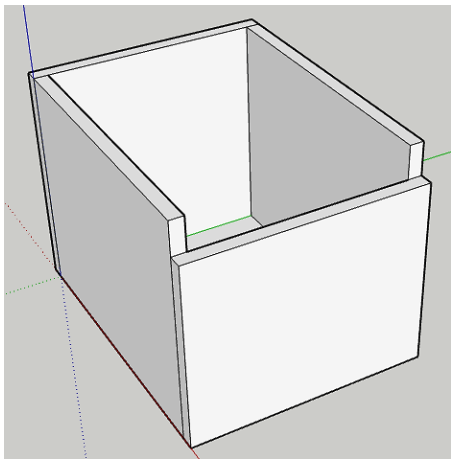
- Open *SketchUp*.
- *Model Info* change *Format*: *Centimeter*, *Display Precision*: 0.0 mm, *Length snapping*: 0.1 mm.
- Select the *Rectangle* tool and input, e.g. 500, 25 mm.
- Select the *Push/Pull* tool and push the model up to a height of 400 mm. Duplicate this object to the right (*Ctrl* + *Move*) at a distance of 400 mm:



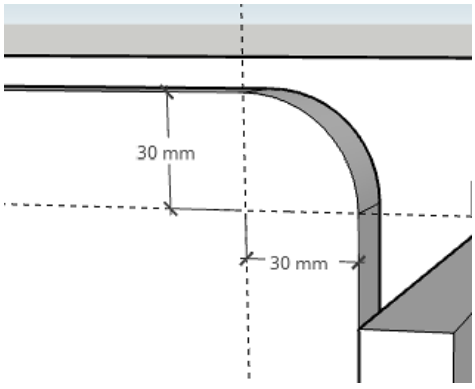
- Connect the two sketched objects by drawing a rectangle (425, 25 mm) on the blue plane and extruding the model to a height of 340 mm:

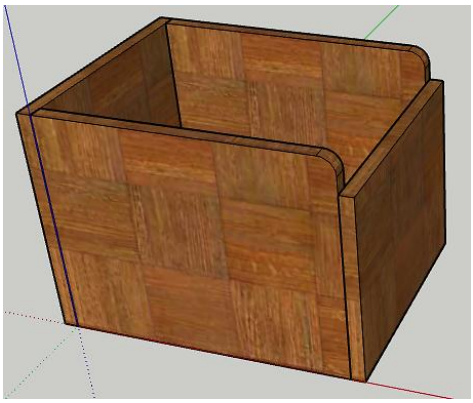


- Repeat the action by drawing the wall of the storage box on the other side (height 400 mm):

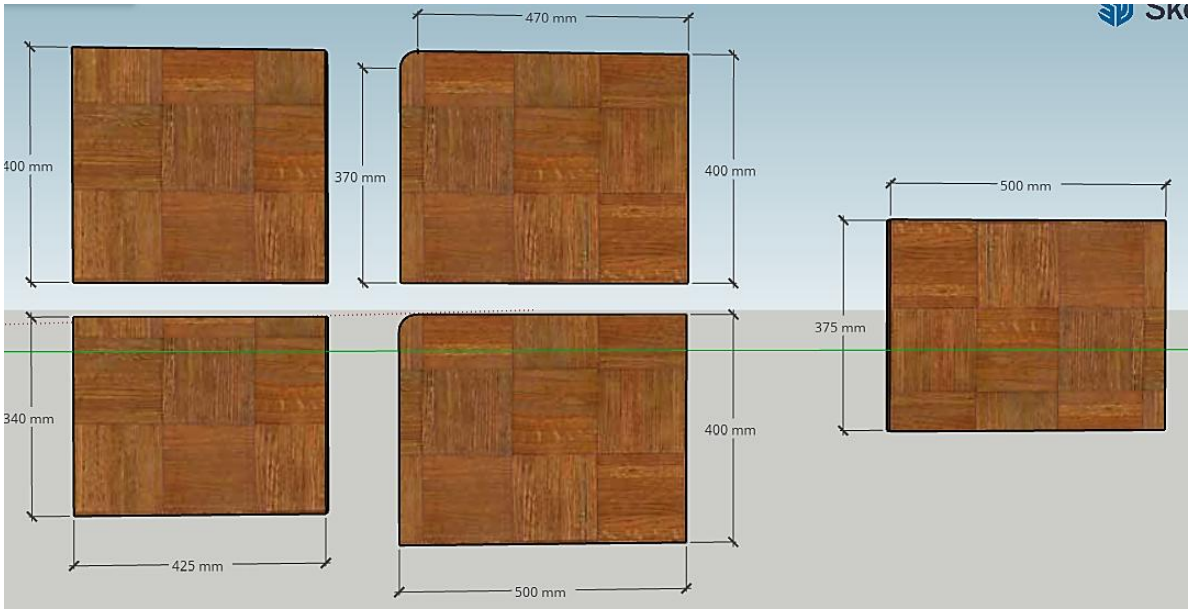


- Using the *2 Point Arc* tool, round off the corners with a radius of 30 mm and remove the unnecessary part. Using ribbon *Materials*/ *Browse*, apply any material of your choice to the storage box, e.g.:





- Create a storage box print-out with all details and necessary dimensions:



Module 2.4. 3D house creation with animation

Learning Objective:
By the end of the module, students will be able to create a 3D house in the online program *SketchUp* and create video animation.

Materials:
- projector;
- devices with internet access (laptops/tablets) for using online resource *SketchUp*[6]
- practical task "3D house creation with animation".

Teaching methods:
- demonstration;
- practical task;
- problem solving;
- learning discussing.

Sources: https://sketchup.trimble.com/en/try-sketchup

1st – 4th practical lesson
Initiation phase (10 min.)
- Updating previous knowledge about various *SketchUp* tools. Discussion with students about various tools and ribbons. Where can I find additional tools?

[6] https://sketchup.trimble.com/en/try-sketchup

Engagement phase (25 min.)

- Use of new information: students work individually with task "3D house creation with animation" practical work.

Reflection phase (5 min.)

- Feedback: What worked? What didn't work?

5th – 8nd practical lesson

Engagement phase (35 min.)

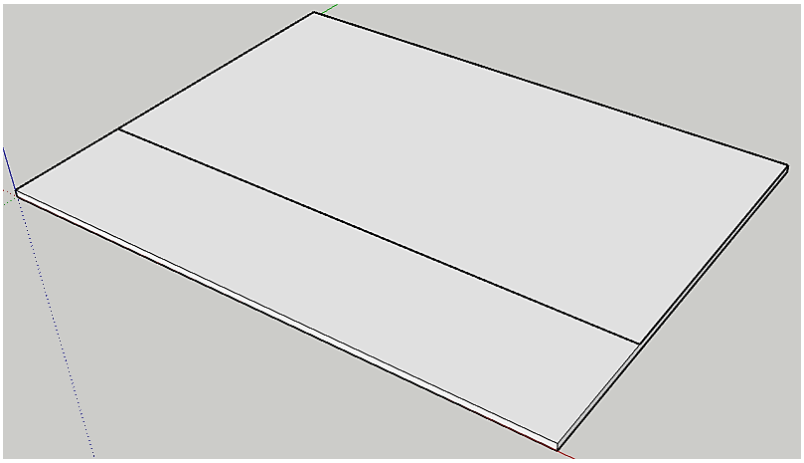
- Updating previous knowledge about various *SketchUp* tools. Discussion with students about various tools and ribbons. How can I create scenes? How can I download multiple objects from the 3D warehouse?
- Use of new information: students continue work individually on the task "3D house creation with animation" - practical work.

Reflection phase (5 min.)

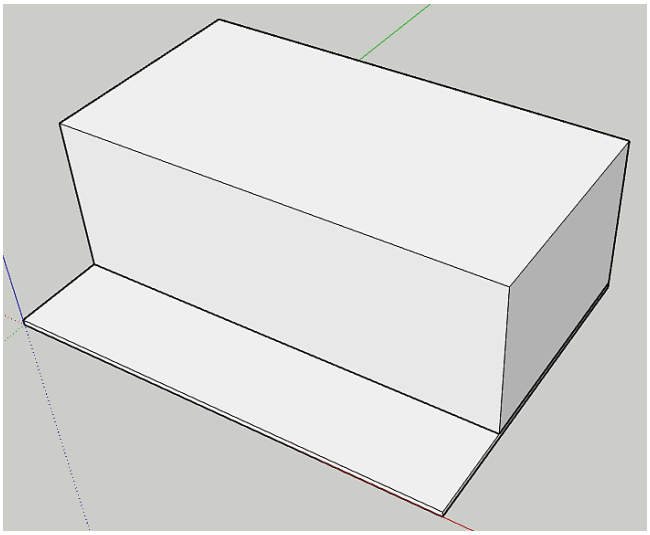
- Feedback: What worked? What didn't work?

Worksheet "3D house creation with animation"

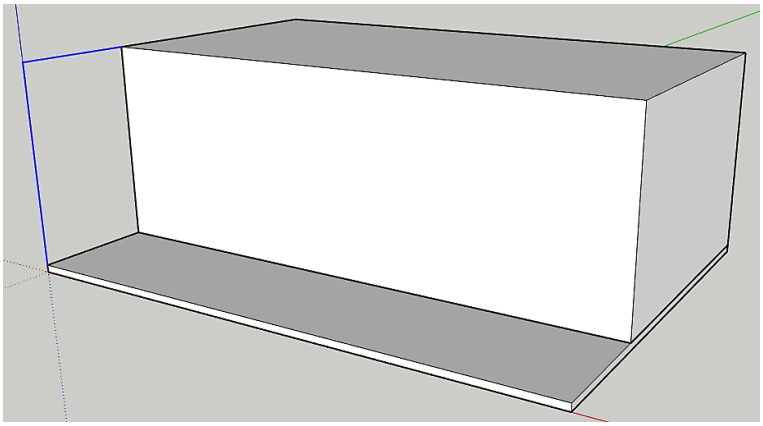
- Open *SketchUp*.
- *Model Info* change *Format*: *Millimeter*, *Display Precision*: 0.0 mm, *Length snapping*: 0.1 mm.
- Select the *Rectangle* tool and input, e.g 8400, 6100 mm.
- Select the *Push/Pull* tool and push the model up to a height of 110 mm.
- The created model needs to be grouped. Highlight the model, then right-click and choose *Make Group*.
- Select the *Rectangle* tool and input, e.g 4300, 8400 mm:



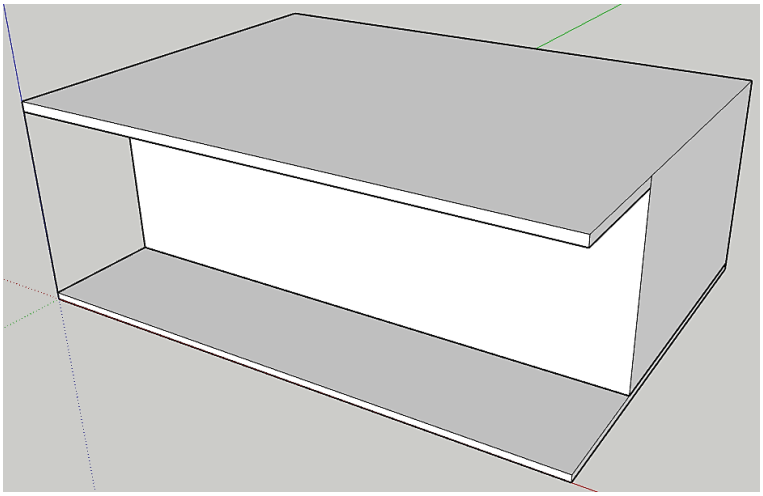
- Select the *Push/Pull* tool and push the model up to a height of 3000 mm.



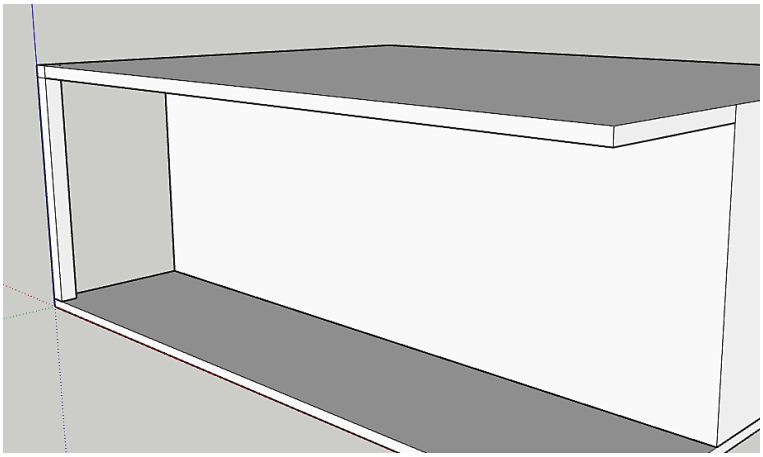
- Using the *Line* tool, sketch the side edge of the house:



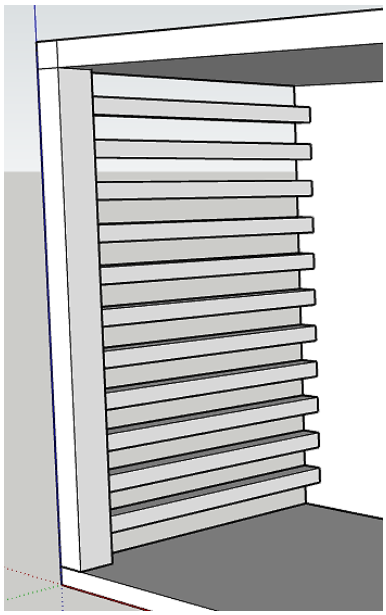
- Using the *Rectangle* tool, sketch the roof of the canopy. Then you can delete unnecessary lines on the roof. Using the *Push/Pull* tool and push the roof of the canopy up to a height of 150 mm:



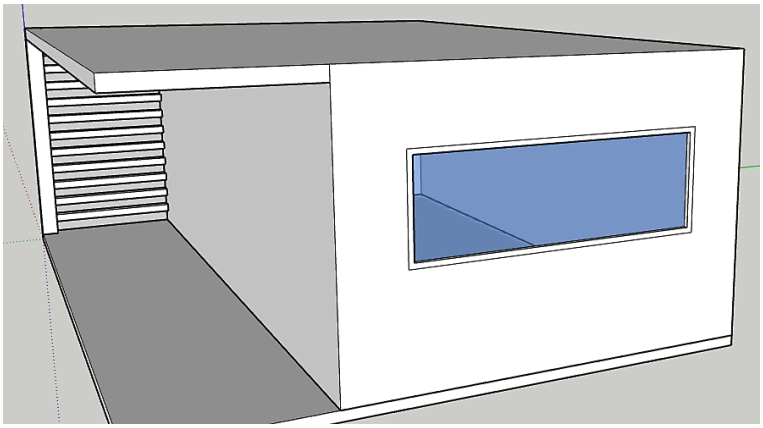
- Using the *Rectangle* tool (200 mm, 200 mm) and the *Push/Pull* tool, to create a pole:



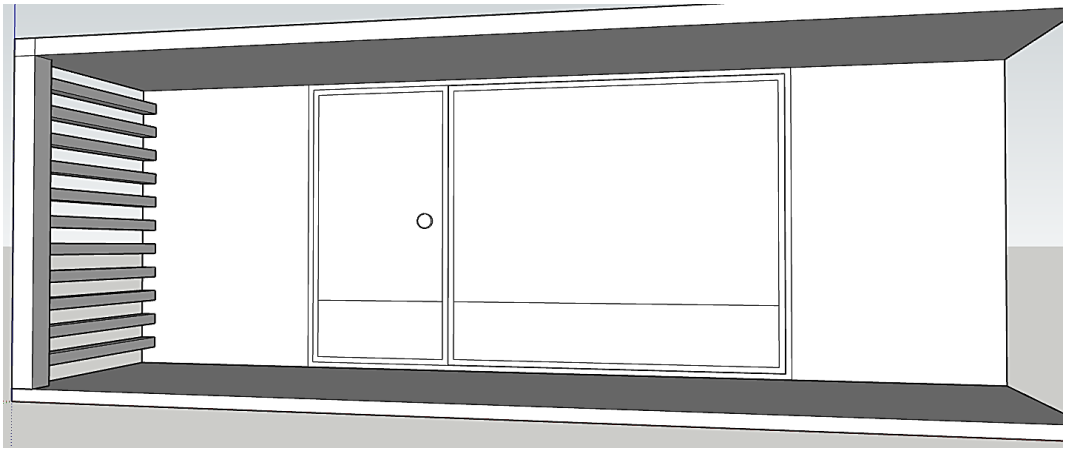
- By duplicating the created pole, create a side wall:



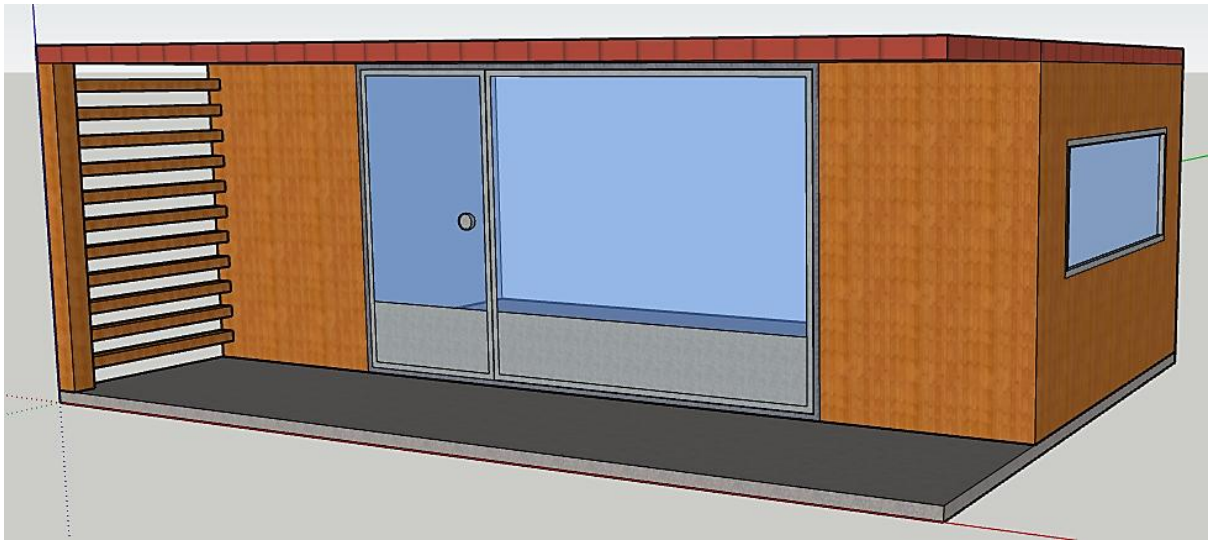
- Using the *Rectangle* tool, the *Offset* tool, and the *Push/Pull* tool, create a window. Then, using the ribbon *Materials*/ *Browse*/ *Glass and Mirrors*, apply the window glass material of your choice, e.g.:



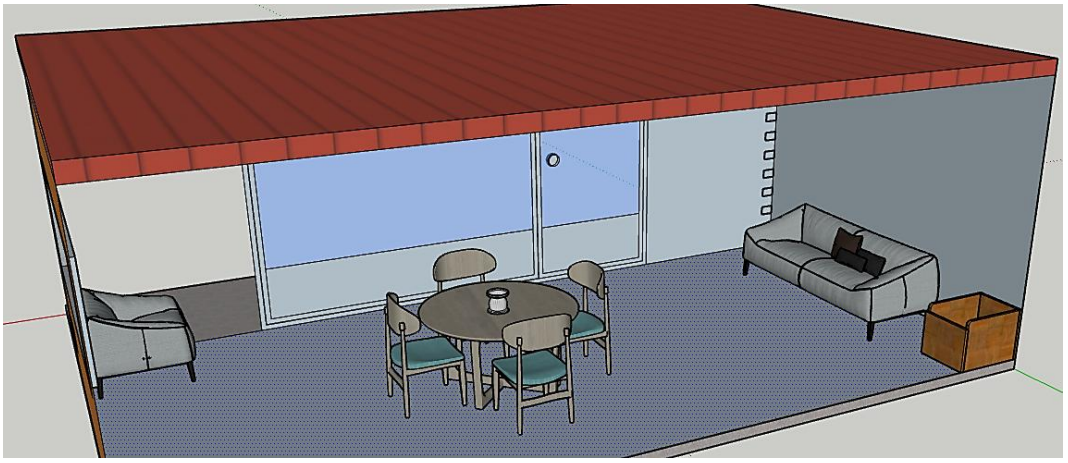
- Using the *Rectangle* tool, the *Offset* tool, *Circle* tool and the *Line* tool, create a door and a panorama window:

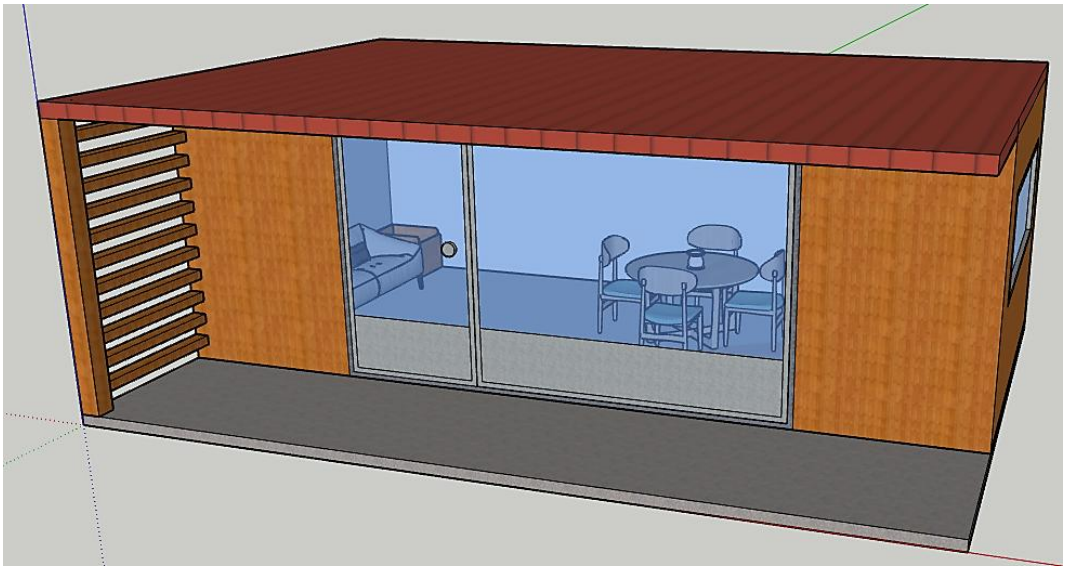


- Using the ribbon *Materials*/ *Browse*, apply any material of your choice to the house, e.g.:



- Using *3D Warehouse*, choose furniture, dishes, and other items you like, and download them. Place the vase and storage box you created in the house.





- Select ribbon *Scenes* and create clips (*Add scene*) by rotating the house model in any way you want.

Module 3. Video creating in your browser (4 hours)

Aim: Develop spatial thinking, creativity, and digital skills in content creation. Understand how to create a video in the free online program *Magisto*.

Outline:

Module 3.1. Video creating with *Magisto*

Time: 4 hours.

Module 3.1. Video creating with *Magisto*

Learning Objective:
By the end of the Module 3.1, students will be able to understand how to create a video in the free online program *Magisto*.

Materials:

- Worksheet "Video creating with *Magisto*".
- Devices with internet access (laptops/tablets) for using online *Magisto*[7].

Teaching methods:

- demonstration;
- problem solving;
- practical task;
- learning discussing.

Sources: https://www.magisto.com/online-video-marketing

1st practical lesson

Initiation phase (10 min.)

- Focusing attention: the teacher shows various video.
- About the goal and results to be achieved: the teacher encourages students to formulate the goal of the lesson.

Engagement phase (30 min.)

- Learning new information: the teacher shows students how to connect to the online resource *Magisto*. Explains how to connect to the free version using an e-mail or *Google, Facebook or Apple* account. The teacher introduces students to the online

[7] https://www.magisto.com/online-video-marketing

resource *Magisto.* Students independently explore the online resource *Magisto.* Compiling information and discussing it with students.

2nd practical lesson

Engagement phase (25 min.)

- Use of new information: students work individually with the worksheet "Video creating with *Magisto*.

Reflection phase (10 min.)

- Feedback: What did you learn about the online resource *Magisto* today? What worked? What didn't work? What else do you plan to learn?

3rd – 4th practical lesson

Engagement phase (25 min.)

- Updating previous knowledge about *Magisto*. Discussion with students about various video creator and AI.
- Use of new information: students continue work individually on the task "Video creating with Magisto" practical work.

Reflection phase (5 min.)

- Feedback: What worked? What didn't work?

Worksheet "Video creating with Magisto"

- Upload your photos files or find and download any image from the free stock image discovery platform, e.g., http://www.unsplash.com or http://www.pixabay.com.
- Then add text, if you want, click *Next*.
- Choose one of unique editing styles from library.
- At the bottom of the window, change the preset colors.
- Choose one of soundtracks from library.
- Click *Preview* and AI powered professional video maker will generate your video.
- Created video can edit or save. You can share with your video (anyone can view, or with link can view, or only you can view).
- You need to download this file.
- Upload your photos files or find and download any image from the free stock image discovery platform, e.g., http://www.unsplash.com or http://www.pixabay.com.
- Then add text.
- Choose one of unique editing styles from the library.
- At the bottom of the window, change the preset colors.
- Upload your music.
- Click *Preview.* Share your video as *Unlisted* (anyone with the link can view it) and share by teacher's e-mail.

- Upload your video files or find and download any video from the free stock video discovery platform, e.g., http://www.pixabay.com.
- Then, add text.
- Choose one of unique editing styles from the library.
- At the bottom of the window, change the preset colors.
- Upload your music.
- Click *Preview.* Share your video as *Unlisted* (anyone with the link can view it) and share by the teacher's e-mail.

Module 4. Introduction to Python and Arduino

Aim: Understand and apply the fundamentals of the Arduino CTC 101 system by exploring its components, setting up the programming environment, and creating the initial programs.

Outline:
Explore the parts and functions of the Arduino CTC 101 kit.
Prepare and configure the programming environment (Processing IDE).
Learn how to create, save, and manage projects on computers.
Write, run, and extend simple programs.

Time/hours: 10 (5 double lessons - 2x40 min)

Module 4.1. Setting up the Programming Environment

Aim: Enable students to become familiar with the Processing and Python library setup, understand the main interface elements, and introduce the concept of coordinate axes in Processing.

Learning Objective:
By the end of the session, students will be able to:

- Understand the purpose of a programming environment for Arduino projects.
- Install Processing under teacher guidance.
- Add the Python library to Processing.
- Navigate and identify key elements of the Processing interface.
- Learn how to create, save, and open projects in Processing.
- Understand the coordinate system in Processing and how it differs from the mathematical system.
- Organize projects by working in a dedicated personal folder.

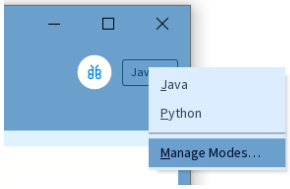
Materials:
Computers with internet access.
Official Processing installation guide: (https://ctc101.arduino.cc/resources?id=31800447)
Worksheet “Processing Interface Identification”.
Worksheet “Processing Coordinates”.

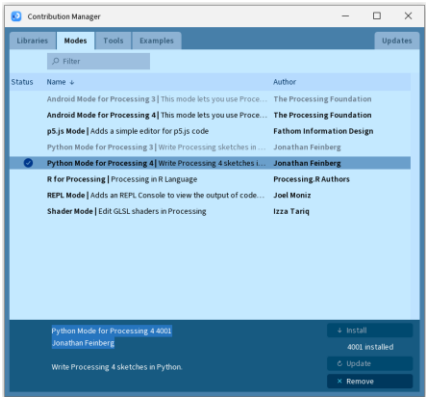
Lesson Outline:

1. Explanation of the necessity of a programming environment for Arduino projects, emphasizing that it enables precise control of hardware components, allows students to implement and test their ideas, and functions as a bridge between the user and the

Arduino's physical components, translating instructions into tangible actions. (5 minutes)

2. Students install Processing under teacher supervision, following the official guide. [1] (5 minutes)
3. Under teacher guidance, students add the Python mode (Python Mode for Processing 4 4001 (Jonathan Feinberg) to Processing to enable programming in Python. (5 minutes)





4. Each student (or students in pairs) creates a dedicated folder on their computers all Processing projects will be saved, ensuring structured work. (5 minutes)
5. Teacher demonstrates how to: (5 minutes)
   - Create a new project.
   - Save it in the student's project folder.
   - Open an existing project.
   - Students then repeat the process on their computers.
6. The teacher explains the Processing interface: menu, editor, run button, and console [2]. (10 minutes)
7. Students complete Worksheet "Processing Interface Identification", labeling the main menu items and interface elements. (10 minutes)
8. Students review their completed worksheets in pairs or groups, discuss any uncertainties, and clarify the purpose and function of each interface element under teacher guidance. (5 minutes)
9. The teacher introduces the measurement units used in Processing [3]: (5 minutes)
   - Pixels – the basic unit for screen coordinates, defining positions and sizes.
   - Radians – used for specifying angles.

Students then discuss why these units are important and how they differ from everyday measurements.

10. The teacher introduces the Processing coordinate system [3]: (10 minutes)
    - The origin (0,0) is in the top-left corner, not the bottom-left as in mathematics.
    - x increases to the right, and y increases downward.
    - Contrast this with the standard mathematical axes, where y increases upward.
11. Students complete Worksheet "Processing Coordinates", which includes a screenshot with several cartoon-style objects. Students estimate and write down the approximate coordinates (x, y) of the center of each object as if they were drawn in Processing. Afterwards, they compare and discuss their answers in small groups, noting any differences in estimation. This activity helps them reinforces understanding of)Processing's coordinate system in an engaging and concrete way.(10 minutes)

Reflection (5 minutes):
Before leaving the class, each student names one Processing interface element and states in a single phrase what it is used for. Students should try not to repeat answers, although some repetition may occur if there are fewer elements than students.

Extension Ideas:

- In addition to labeling the interface elements in the worksheet “Processing Interface Identification”, ask students to describe in their own words the function of each element next to its name.
- Students may work in pairs to compare their descriptions and discuss any differences in understanding.

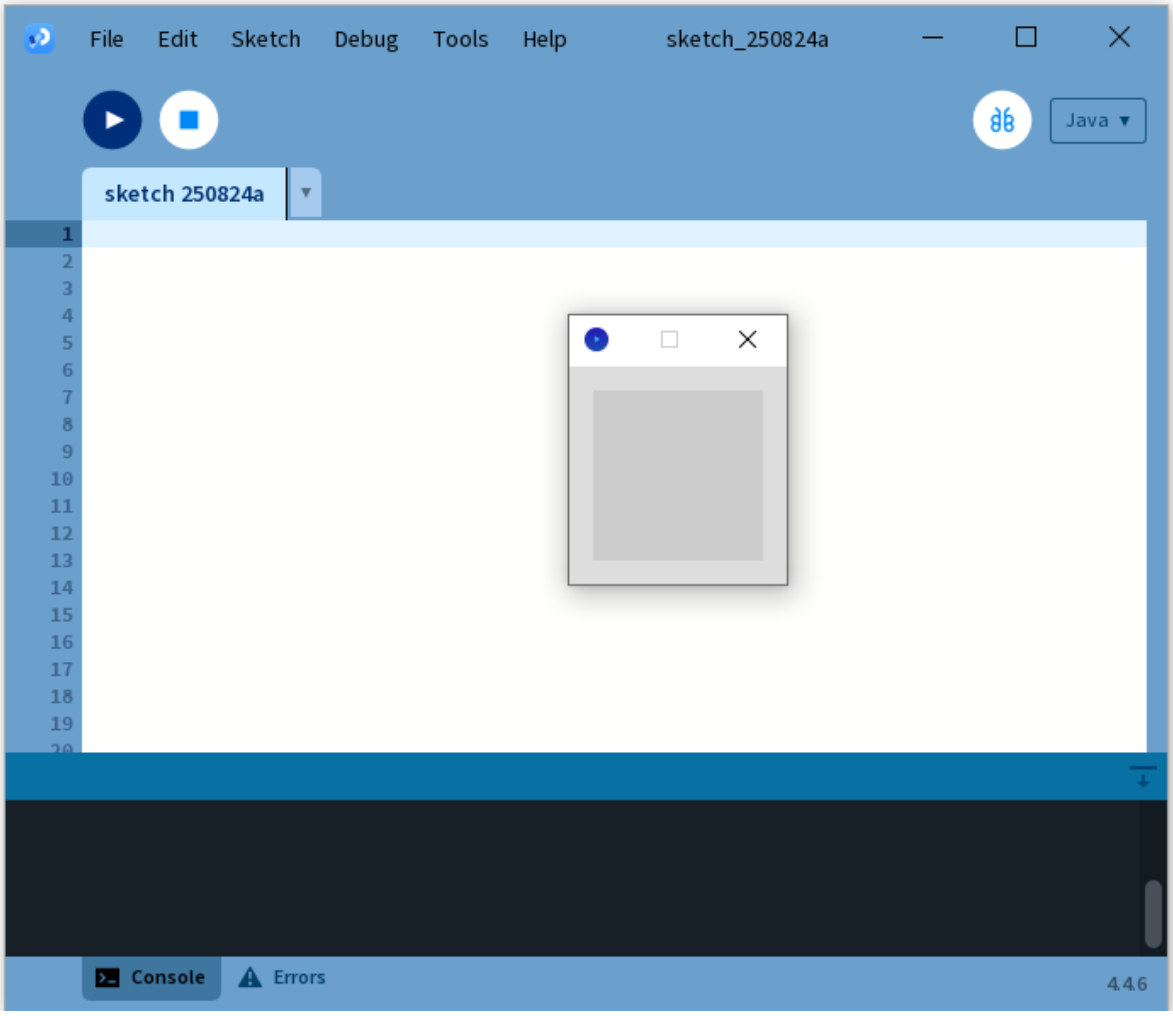
Sources:
[1] https://ctc101.arduino.cc/resources?id=31800447
[2] https://ctc101.arduino.cc/ctc101/module/module-1/lesson/processing
[3] https://ctc101.arduino.cc/ctc101/module/module-1/lesson/screens-and-pixels

Worksheet "Processing Interface Identification"

Become familiar with the Processing interface. Examine the provided screenshot of the Processing and label the main elements. Check the box in the list once you have identified each element.

On the screenshot of the Processing below, identify and label the following elements:

☐ Toolbar
☐ Editor Window
☐ Console
☐ Run Button
☐ Stop Button
☐ Tabs (for multiple files)
☐ Errors (Error Area)
☐ Programming Language Selector (dropdown menu)
☐ Output Window
☐ Project Name



Worksheet "Processing Coordinates"

Key Information
In Processing, the coordinate system is different from the usual mathematical system.

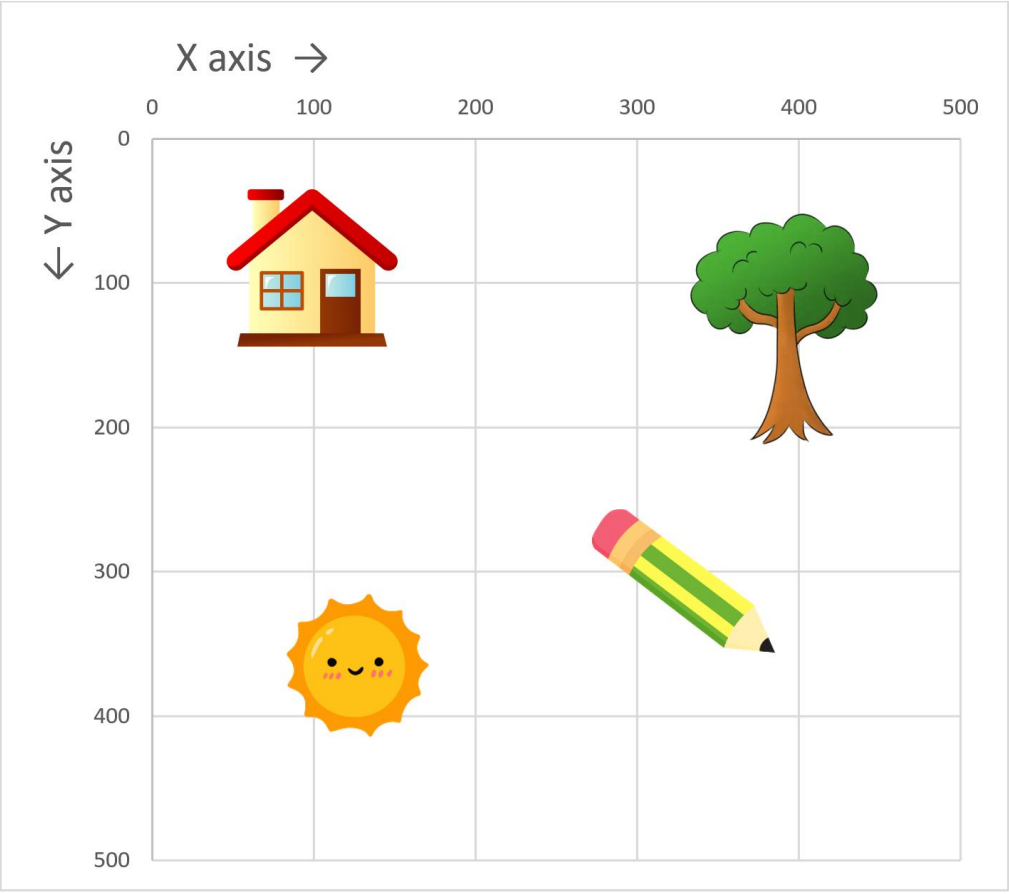
- The origin (0,0) is in the top-left corner of the output window.
- The x-axis increases to the right. ⟶
- The y-axis increases downward. ↓

Example:

- point(0, 0)- top-left corner.
- point(100, 50)- 100 pixels to the right, 50 pixels down from the top-left corner.

Tasks
Look at the provided screenshot with simple objects. For each object, write down the approximate coordinates (x, y) of its center.

Object	Your Estimated Coordinates (x, y)
sun	
tree	
house	
pencil	



How is this coordinate system different from the one you've seen in math class? (Write a short answer.)

_____

_____

_____

If you finish early, draw one more object of your choice (a cloud, car, robot, heart, or any other). Mark the coordinates of the center of your drawn object (x, y). Then ask a partner to estimate the center coordinates of your object and compare your results.

Module 4.2. First Steps in Programming with Graphics

Aim: Introduce students to programming using Python in Processing by creating simple graphics. Students will learn how code translates into visual output and will understand basic programming concepts such as drawing commands, sequencing, and comments.

Learning Objective:
By the end of the session, students will be able to:

- Understand how Python code in Processing creates graphics.
- Use the functions size(), line(), rect(), and ellipse().
- Use fill(R, G, B) to set the fill color of shapes and stroke(R, G, B) to set the outline color.
- Understand and use comments in code to annotate programs.
- Apply sequencing to create drawings and simple patterns.

Materials:
Computers with Processing and Python library installed.
Projector or board for teacher demonstrations.
Worksheet “Graphics Programming”.

Lesson Outline:

1. The teacher explains drawing commands and comments (#) in Python code, showing examples on the board or projector. (5 minutes)
2. Introduce the syntax: size(width, height) and give a simple example such as (300, 300). Explain how size() sets the "canvas" (Output Window) for all drawings. [1] [2] (5 minutes)
3. The teacher explains how to set colours using fill(R, G, B) for a shape’s interior and stroke(R, G, B) for its outline. Mentions briefly that R, G, B values range from 0 to 255. Demonstrates setting up the output window (size()), using a line (line()), rectangle (rect()), and ellipse (ellipse()) functions using Python in Processing, adding comments to each function. Highlights the role of parameters (x, y, width, height) in defining shape properties. [3][4][5][6] (15 minutes)
4. Students replicate teacher’s examples on their computers and experiment with changing numbers in the functions to observe results. The teacher circulates and provides help as needed. (15 minutes)
5. Students complete tasks from the worksheet “Graphics Programming”. (25 minutes)
6. Selected students present their solutions using the projector or share them with classmates. The teacher highlights creative approaches and common mistakes. (10 minutes)

Reflection (5 minutes):
Students discuss the following guiding questions, either in pairs or as a class:

- What did you notice about how the code translates into visuals?
- Which part of writing the code was easy for you?
- Which part was challenging, and why?

- Discuss how using different fill and stroke colours changed the appearance of the shapes.

Extension Ideas:

- Challenge students to combine shapes into more complex figures (for example, a car, a tree, a robot face or flower).
- Ask students to add comments to their code explaining each line.
- Encourage students to explore colour functions (fill(), stroke()) if time allows.

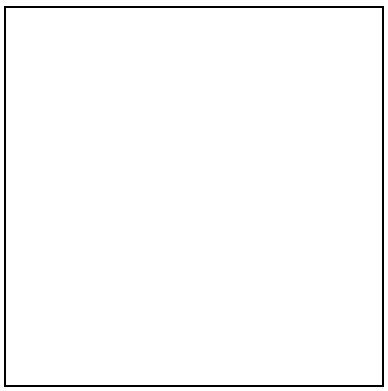
Sources:

[1] https://processing.org/reference/size_.html
[2] https://ctc101.arduino.cc/ctc101/module/module-1/lesson/setup-and-draw
[3] https://ctc101.arduino.cc/ctc101/module/module-1/lesson/line
[4] https://processing.org/reference/ellipse_.html
[5] https://processing.org/reference/rect_.html
[6] https://processing.org/reference/fill_.html

Worksheet "Graphics Programming"

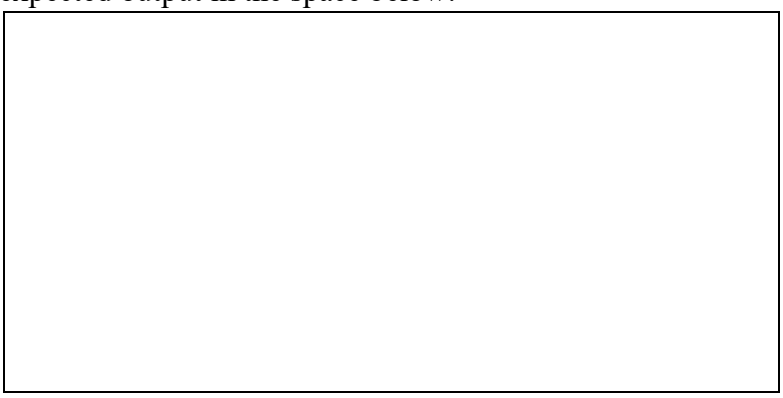
Task 1. Simple Drawing

1. Write a program that opens an output window of 500 × 500 pixels (use size()).
2. Draw a rectangle in the center of the window (use rect()).
3. Draw a circle (an ellipse with equal width and height) above the rectangle (use ellipse()).
4. Draw a diagonal line across the screen (use line()).
5. Sketch your output:



Task 2. Triangle Challenge

1. Write a program that opens an output window of any size you choose (use size()).
2. Draw an isosceles triangle (a triangle with two sides of equal length) in the center of the window using three line() commands. Make sure all lines connect to form a closed shape.
3. Draw a right-angled triangle (a triangle with one 90° angle) in another part of the window using three line() commands.
4. Add comments to explain each line of your code.
5. Sketch your expected output in the space below:



Task 3. Explore the arc() Function

About arc():

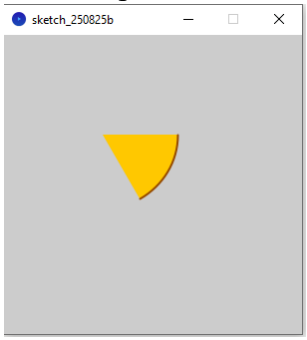
The arc() function in Processing draws a section of an ellipse (a curved line).

Syntax:

- arc(x, y, width, height, start, stop);
- x, y – coordinates of the ellipse center;
- width, height – size of the ellipse;
- start, stop – angles in radians defining the section to draw.

Important for students:
Angles in arc() are always in radians, not degrees. To make it easier, you can use the function radians(degrees) to convert degrees to radians automatically.

Example:
Draw a “pizza slice” arc. Result:



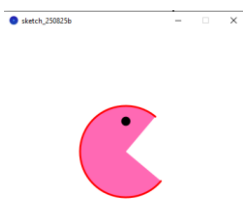
Python code:

```
size(300, 300)
fill(255, 200, 0)          # Yellow-orange fill for the pizza
stroke(150, 75, 0)         # Brown outline
strokeWeight(2)             # Sets the thickness of the outline (border) of shapes
to 2 pixels
arc(100, 100, 150, 150, radians(0), radians(60))  # Draw a slice (60°)
```

The task: Pac-Man

1. Open an output window of any size you choose (use size()).
2. Draw Pac-Man using arc(). Choose start and stop angles so that it looks like an open mouth.
3. Draw Pac-Man’s eye using ellipse().
4. Add comments to explain each line of code.
5. Optional: Use fill(), stroke(), and strokeWeight() to personalize Pac-Man’s colour and outline.

Example:
Window size: 400 × 400 pixels
Pac-Man body: bright pink, red outline, outline thickness 3.
Eye: black.
Result:



Additional information about arc() function can be found here:
https://py.processing.org/reference/arc

Color Reminder- RGB Format
Colors in Processing are specified using the RGB format: fill(R, G, B) or stroke(R, G, B)
R, G, B stand for:

- R = Red (red component of the colour);
- G = Green (green component of the colour);
- B = Blue (blue component of the colour).

Each value can range from 0 to 255:

- 0 = none of that colour;
- 255 = full intensity of that colour.

The combination of R, G, and B determines the final colour:

- fill(255, 0, 0)- bright red;
- fill(0, 255, 0)- bright green;
- fill(0, 0, 255)- bright blue;
- fill(255, 255, 0)- yellow (red + green);
- fill(0, 255, 255)- cyan (green + blue).

Tip: By changing the values of R, G, and B, you can create millions of different colours for your shapes.

Module 4.3. Getting Started with Variables

Aim: Understand and apply variables using Python in Processing and learn how setup() and draw() control the program flow.

Learning Objective:
By the end of the session, students will be able to:

- Understand what a variable is and how it can store and change values.
- Learn the difference between setup() (runs once) and draw() (runs continuously).
- Use variables to control visual elements (e.g., positions, sizes, or colours).
- Begin developing structured code.

Materials:
Computers with Processing and Python library installed.
Projector or board for teacher demonstrations.
Worksheet “Variables in Action”.
Worksheet “Setup vs Draw Challenge”.

Lesson Outline:

1. The teacher introduces the concept of variables, comparing them to "boxes that store information", and explains their role in programming.[1] (10 minutes)
2. The teacher demonstrates a simple example: a circle whose x-position is controlled by a variable. Shows how changing the variable makes the circle move. [1] (10 minutes)
3. Students replicate the moving circle example on their computers. They experiment with speed and direction by changing the variable values (makes the circle move). (15 minutes)
4. The teacher explains the structure of setup() and draw(), emphasizing that setup() runs once for initialization and draw() repeats continuously for animation. [3][4][5][6] (10 minutes)
5. Students complete Worksheet “Variables in Action”, filling in code snippets and answering short questions. (15 minutes)
6. The teacher introduces a small challenge: place print("Hello") in both setup() and draw(). Students predict the difference, then test and record results in Worksheet “Setup vs Draw Challenge”. (15 minutes)

Reflection (5 minutes):
As a class, students briefly share one example of how they used a variable in their drawing or what they discovered about how setup() and draw() work differently. The teacher encourages students to use correct terminology (variable, loop, initialization) while explaining their insights

Extension Ideas:

- In Worksheet “Variables in Action”, students can expand their sketches by adding more variables (e.g., colour or size) to make objects move or change dynamically.

- In Worksheet “Setup vs Draw Challenge”, students can modify the example to include a variable that changes within draw() (for instance, gradually moving a circle across the screen) to observe how repeated execution creates animation.
- Encourage advanced students to design a short animation that combines variables, loops, and colour changes.

Sources:
[1] https://processing.org/examples/variables.html
[2] https://ctc101.arduino.cc/ctc101/module/module-1/lesson/variables
[3] https://processing.org/reference/draw_.html
[4] https://processing.org/examples/setupdraw.html
[5] https://processing.org/reference/setup_.html
[6] https://ctc101.arduino.cc/ctc101/module/module-1/lesson/setup-and-draw

Worksheet "Variables in Action"

Task 1. Example
Carefully review the example provided below:

```
x = 50
y = 200
speed = 3

def setup():
    size(400, 400)
    background(255)

def draw():
    global x, speed
    background(255)
    fill(255, 100, 100)
    ellipse(x, y, 50, 50)
    x = x + speed

    if x > 375 or x < 25:
        speed = -speed
```

Answer questions:

1. What causes the ball to change direction?

2. What would happen if you removed the line background (255)?

3. Which variables control the motion and which control the appearance?

Task 2. Create Your Own Animation
Create your own sketch using at least two variables.
Examples of ideas:

- A spaceship flying across the sky.
- A robot blinking its eyes.
- A balloon moving upward and changing colour.
- You may also create your own original idea, as long as it uses at least two variables to control movement, colour, or other properties.

Describe your idea below before coding it:

My project idea:

Write the names of the variables you plan to use in your program (for example: x_position, y_position, colour_change) and briefly describe what each variable controls:

Worksheet "Setup vs Draw Challenge"

Experiment with setup() and draw() to understand their timing, repetition, and role in Processing programs.

Task 1. Predict the Output

Study the following code and write what you think will happen before running it:

```
def setup():
    print("Setup runs once")

def draw():
    print("Draw runs many times")
```

Your prediction:

Task 2. Run and Record

Run the program and check the console output. Write how many times each message appears.

Function	Message Printed	How Many Times?
setup()		
draw()		

Task 3. Change and Try

Change the code to include your own text:

```
def setup():
    print("Welcome to Processing!")

def draw():
    print("Learning is fun!")
```

Run it again and describe what happens:

Task 4. Explain What You Learned

In one or two sentences, describe when you would use each function.

setup():

draw():

Module 4.4. Creative Project: From Code to Game

Aim: Apply knowledge of variables and program structure to design and implement a simple interactive project using Processing with Python.

Learning Objective:
By the end of the session, students will be able to:

- Reinforce understanding of variables by applying them to moving objects.
- Use setup() and draw() to create dynamic and continuous animations.
- Experiment with interactivity.
- Develop creativity and problem-solving through a guided project.

Materials:
Computers with Processing and Python library installed.
Projector or board for student demonstrations.
Worksheet "Creative Coding Challenge: Bouncing Ball".

Lesson Outline:

1. The teacher reviews what variables are and how setup() and draw() work. Introduces the concept of animation using a moving object that changes position over time. Shows a simple example of a ball moving horizontally using a variable for position. [1][2][3][4](10 minutes)
2. The teacher demonstrates how to make the ball bounce off the edges of the window by reversing direction when it reaches the border. Example code is displayed and explained step-by-step, highlighting the role of variables for position, speed, and direction. [5] (10 minutes)
3. Students reproduce the example on their computers, experimenting with (The teacher circulates and supports students as needed.): (20 minutes)
    - Different window sizes (size(width, height))
    - Changing the ball's colour (fill(R, G, B))
    - Adjusting speed or direction.
4. Students complete Worksheet "Creative Coding Challenge: Bouncing Ball", writing a program that makes a ball move and bounce inside the window. (30 minutes)
5. Selected students present their projects using the projector. The teacher highlights creativity, correct use of variables, and problem-solving strategies. (10 minutes)

Reflection (5 minutes):
In pairs, students briefly discuss what made their ball move and bounce. Each student writes one short sentence on their worksheet answering: "How did variables help control your animation?"
Extension Ideas:

- Add interactivity to the program: make the ball change direction when a key is pressed or colour when clicked.
- Create multiple balls moving at different speeds using lists of variables.
- Encourage advanced students to turn the bouncing ball into a short mini-game (for example, "catch the ball with a paddle").

- Experiment with sound or background colour changes for additional engagement.

Sources:
[1] https://processing.org/reference/draw_.html
[2] https://processing.org/reference/setup_.html
[3] https://processing.org/examples/setupdraw.html
[4] https://ctc101.arduino.cc/ctc101/module/module-1/lesson/setup-and-draw
[5] https://processing.org/examples/bounce.html

Worksheet "Creative Coding Challenge: Bouncing Ball"

Use your knowledge of variables, setup(), and draw() to create an animated ball that moves and bounces around the screen.

Task 1. Analyze the Example

Carefully review the example provided below:

```
def setup():
    size(400, 400)
    global x, y, speed_x, speed_y
    x = 200
    y = 200
    speed_x = 3
    speed_y = 2

def draw():
    background(255)
    fill(255, 0, 150)
    ellipse(x, y, 40, 40)

    global x, y, speed_x, speed_y
    x = x + speed_x
    y = y + speed_y

    if x > width or x < 0:
        speed_x = -speed_x
    if y > height or y < 0:
        speed_y = -speed_y
```

Answer question:

What happens when the ball reaches the edge of the window??

_____

_____

Task 2. Modify and Experiment

Change at least one aspect of the animation:

- Ball color (fill(R, G, B))
- Ball speed (speed_x, speed_y)
- Ball size
- Window size

Describe what you changed:

_____

_____

_____

Task 3. Create Your Own Bouncing Ball

Now create your own version of the bouncing ball!

Your program should:

- Use at least three variables.
- Make the ball move and bounce inside the window.

- Optionally include colour change, interactivity, or unique motion.

Write your idea here:

Module 4.5. Introduction to Arduino Components

Learning Objective:
By the end of the session, students will be able to:

- Identify and describe the Arduino 101 board, Education Shield, breadboard, sensors, and actuators.
- Explain the role of each major component in building circuits and robots.

Materials:
Arduino CTC 101 Program – FULL (AKX00002) toolbox.
Worksheet "Arduino CTC 101 Component Identification".

Lesson Outline:

1. Introduction to Arduino, main components. Explanation of how Arduino differs from LEGO robotics and other robot platforms familiar to students. [1] (10 minutes)
2. Explanation of main Arduino 101 components and its functions: shields, boards, resistors, modules, speakers, wires etc. [1][2][3][4][5][6][7][8][9] (20 minutes)
3. Using the worksheet "Arduino CTC 101 Component Identification", students unbox the Arduino CTC 101 kit and identify each component. As an optional challenge, only the component names are given, requiring students to draw a schematic representation of each component in the "Picture" column. This activity reinforces component recognition and understanding of their functions. (30 minutes)
4. Conduct a class discussion in which students consider and discuss, "Which component do you believe will be most essential for building a robot, and why?" (10 minutes)

Reflection (10 minutes):
Conduct a class discussion in which students in which students name one component and explain its function. Students are not allowed to repeat.

Extension Ideas:

- Instead of providing the images in the worksheet, give students only the component names and function description.
- Students are required to draw a simple schematic or representation of each component in the "Picture" column of the worksheet "Arduino CTC 101 Component Identification".
- Encourage students to annotate their drawings with any relevant details, such as color, shape, or unique features.
- After completing the drawings, students can compare their sketches with the actual components to discuss similarities and differences.

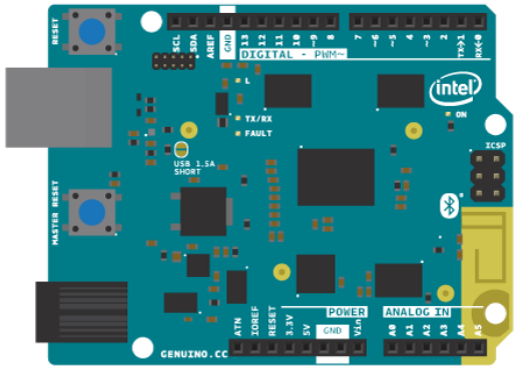
Sources:
[1] https://docs.arduino.cc/hardware/
[2] https://ctc101.arduino.cc/ctc101/module/module-1/lesson/processing
[3] https://ctc101.arduino.cc/ctc101/module/module-2/lesson/what-is-the-ctc-board
[4] https://ctc101.arduino.cc/ctc101/module/module-3/lesson/reading-analog-signals
[5] https://ctc101.arduino.cc/ctc101/module/module-3/lesson/writing-analog-signals

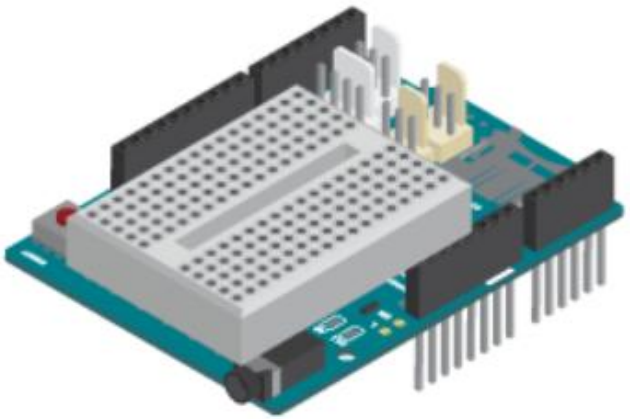
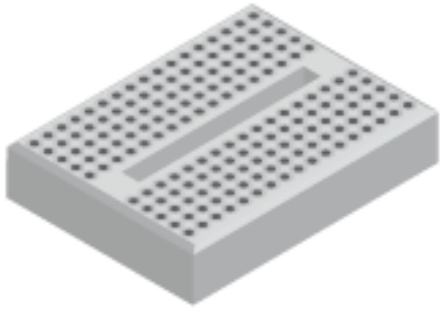
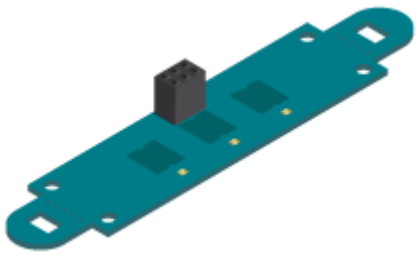
[6] https://ctc101.arduino.cc/ctc101/module/module-3/lesson/light-sensor
[7] https://ctc101.arduino.cc/ctc101/module/module-3/lesson/serial-port
[8] https://ctc101.arduino.cc/ctc101/module/module-4/lesson/types-of-motors
[9] https://ctc101.arduino.cc/ctc101/module/module-5/lesson/what-is-an-imu

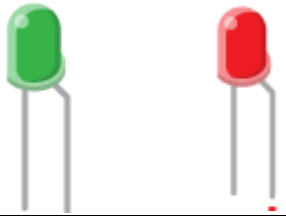
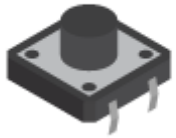
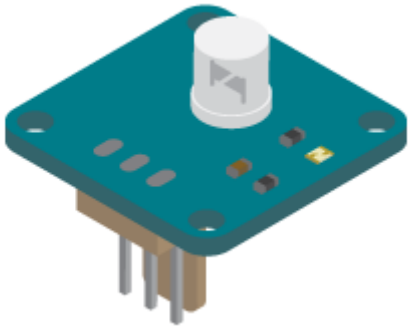
Worksheet "Arduino CTC 101 Component Identification"

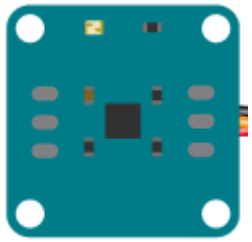
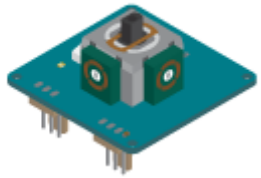
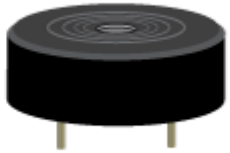
Task 1
Unbox your Arduino CTC 101 kit and identify each component using the table below. Fill "Notes" column if:

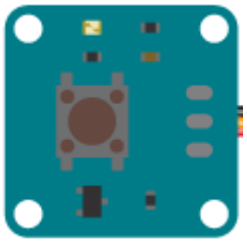
- The component colour is different from the picture,
- You had difficulties finding the component,
- Any other comment or observation about the component.

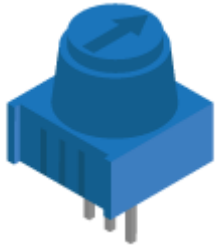
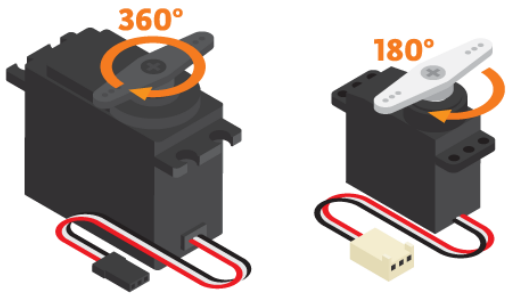
Component Name	Picture	Function	Notes	Found ☒
Arduino 101 Board		Main microcontroller with Bluetooth end IMU (An Inertial Measurement Unit)		☐

Component Name	Picture	Function	Notes	Found ☒
Education Shield		Extends board, organizes components, SD/audio		☐
Breadboard		Build circuits without soldering		☐
IR Array		Consists of 3 IR (Infrared) sensors		☐

Component Name	Picture	Function	Notes	Found ☒
LED		Light indicator		☐
Resistor		Limit current		☐
Push Button		Manual on/off control		☐
Light Sensor Module		Detects light intensity		☐

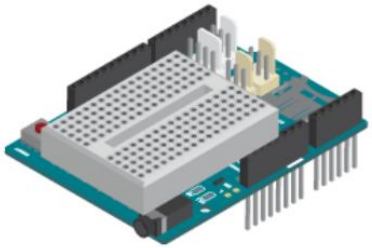
Component Name	Picture	Function	Notes	Found ☒
Tilt Sensor Module		Detects orientation		☐
Joystick Modulino		Provides 2-axis input		☐
Piezo Speaker		Outputs sound via audio connector		☐
Speaker		Outputs sound		☐

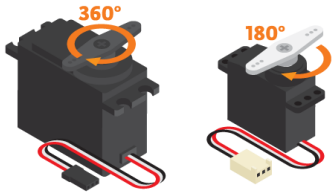
Component Name	Picture	Function	Notes	Found ☒
USB Cable		Connect the board to the computer		☐
Jumper Wires		Connect components		☐
Module Connector Wire		Connect external hardware		☐
Push Button TaModule		Manual on/off control		☐

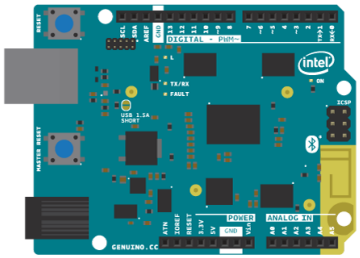
Component Name	Picture	Function	Notes	Found ☒
Potentiometer		Control a project by providing a variable analog input signal		☐
Web Camera		Capture and transmit video		☐
Servo Motor		Precise movement control		☐

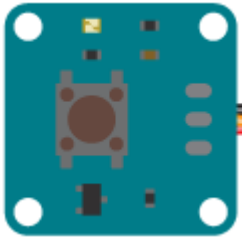
Task 2

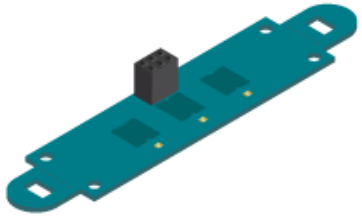
Write down the appropriate component name from the given list under its picture: Arduino 101 Board, Education Shield, Breadboard, IR Array, LED, Resistor, Push Button, Light Sensor Module, LED, Tilt Sensor Module, Joystick Modulino, Piezo Speaker, Speaker, USB Cable, Jumper Wires, Module Connector Wire, Push Button Module, Potentiometer, Web Camera, Servo Motor.

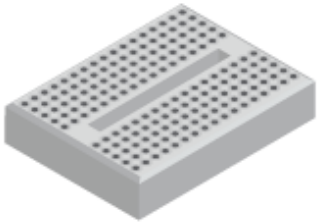


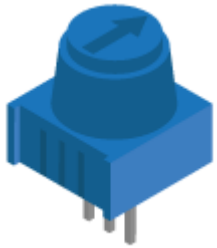


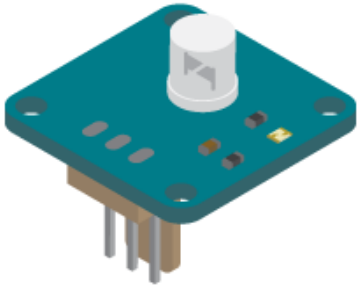


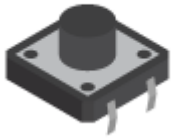


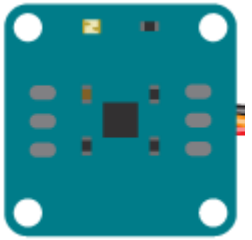


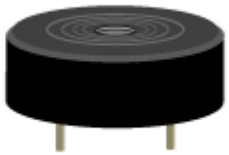


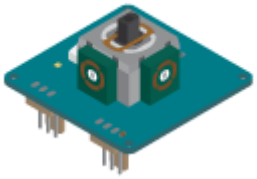












Module 5. Basics of Control Board Programming

Aim: Understand the structure of control board. Learn the basics of digital technologies to control digital actuators and read digital sensors.

Outline:

Module 2.1. Installation and connection of the board in the IDE (Integrated Development Environment)
Module 2.2. Programming of LED blinking and the sound producing using Piezo Speaker
Module 2.3. Project development: building and playing small electronic game that simulates sport games

Time/hours: 6 or 8 (3 or 4 double lessons - 2x40 min)

Module 5.1. Installation and connection of the board in the IDE (Integrated Development Environment)

Learning Objective:
By the end of the lesson, students will be able to understand what is the control board and its structure, install and connect control board it to the IDE, work with IDE using tool bar and message area, upload and compile programmes, understand digital signals.

Materials:
- the 101 control board
- Internet connection
- USB cable

Lesson Outline:
1. Explain students the meaning and structure of the control board using Arduino 101 board as an example. Mention other examples [1, 4] (10 minutes).
2. Introduce students what components are necessary to connect the board (2 minutes).
3. Provide students the instruction how to install IDE. Let them install IDE [2 or Task1] (20 minutes).
4. Introduce students to the IDE, i.e. show the feature of tool bar and message area [1]. Ask students to fulfil the Task3 (10 minutes).
5. Connect the board to the IDE by following instructions given in the Task2 or [3] (20 minutes).
6. Explain the Education shield capabilities [5] and digital signals [6] (10 minutes).
7. Reflection (8 minutes)
– Complete Tasks 1, 2, 3 and 4.
– Students summarize what they learned and share personal insights or preferences regarding control boards, their types and Education shield.

Sources:
[1] https://ctc101.arduino.cc/ctc101/module/module-2/lesson/what-is-the-ctc-board
[2] https://ctc101.arduino.cc/resources?id=31820576
[3] https://ctc101.arduino.cc/resources?id=31818016
[4] https://ctc101.arduino.cc/resources?id=32381557

[5] https://ctc101.arduino.cc/resources?id=31800414
[6] https://ctc101-dev.arduino.cc/ctc101/module/module-2/lesson/digital-signals

Worksheet "Exploring IDE"

Task 1. Installation of IDE

Windows - with administrator rights

1. Go to this page to find the latest version of the IDE: **http://arduino.cc/en/Main/Software**
2. Scroll down and click on 'Windows Installer' to download.
3. When the download has finished, open the downloaded file.
4. Click 'I agree' to agree to the license agreement.
5. Choose all components to be installed and click 'Next'.
6. Remember the destination folder, and click 'Install'.
7. When the installation has finished, click 'Close'.

Windows - without administrator rights

1. Go to this page to find the latest version of the IDE: **http://arduino.cc/en/Main/Software**
2. Scroll down and click on 'Windows ZIP file (for non-administrator install)' to download.
3. When the download has finished, unzip the file to the computer hard drive.

Drivers

1. Connect the control board to your computer using a USB cable.
2. Wait for Windows to begin its driver installation process. After a few moments, the process will fail, despite its best efforts.
3. Open up the Control Panel.
4. Navigate to System and Security. Next, click on System. Once the System window is up, open the Device Manager.
5. Look under Ports (COM & LPT). You should see an open port with the name of your control board and a port to it locking like this:'(COMxx)'. If there is no COM & LPT section, look under 'Other Devices' for 'Unknown Device'.
6. Right click onthe item with the name of your control board and choose the 'Update Driver Software' option.
7. Next, choose the 'Browse my computer for Driver software' option.
8. Finally, navigate to and select the driver file named 'arduino.inf", located in the 'Drivers' folder of the Arduino Software download (not the 'FTDI USB Drivers' sub-directory). If you are using an old version of the IDE (1.0.3 or older), choose the Uno driver file named 'Arduino UNO.inf"
9. Windows will finish up the driver installation from there.

Mac OS X

1. Go to this page to find the latest version of the Arduino IDE: **http://arduino.cc/en/Main/Software**
2. Scroll down and click on 'Mac OS X' to download.
3. When the download has finished, double click the .zip file.
4. Copy the Arduino application into the Applications folder.

Linux - with administrator rights

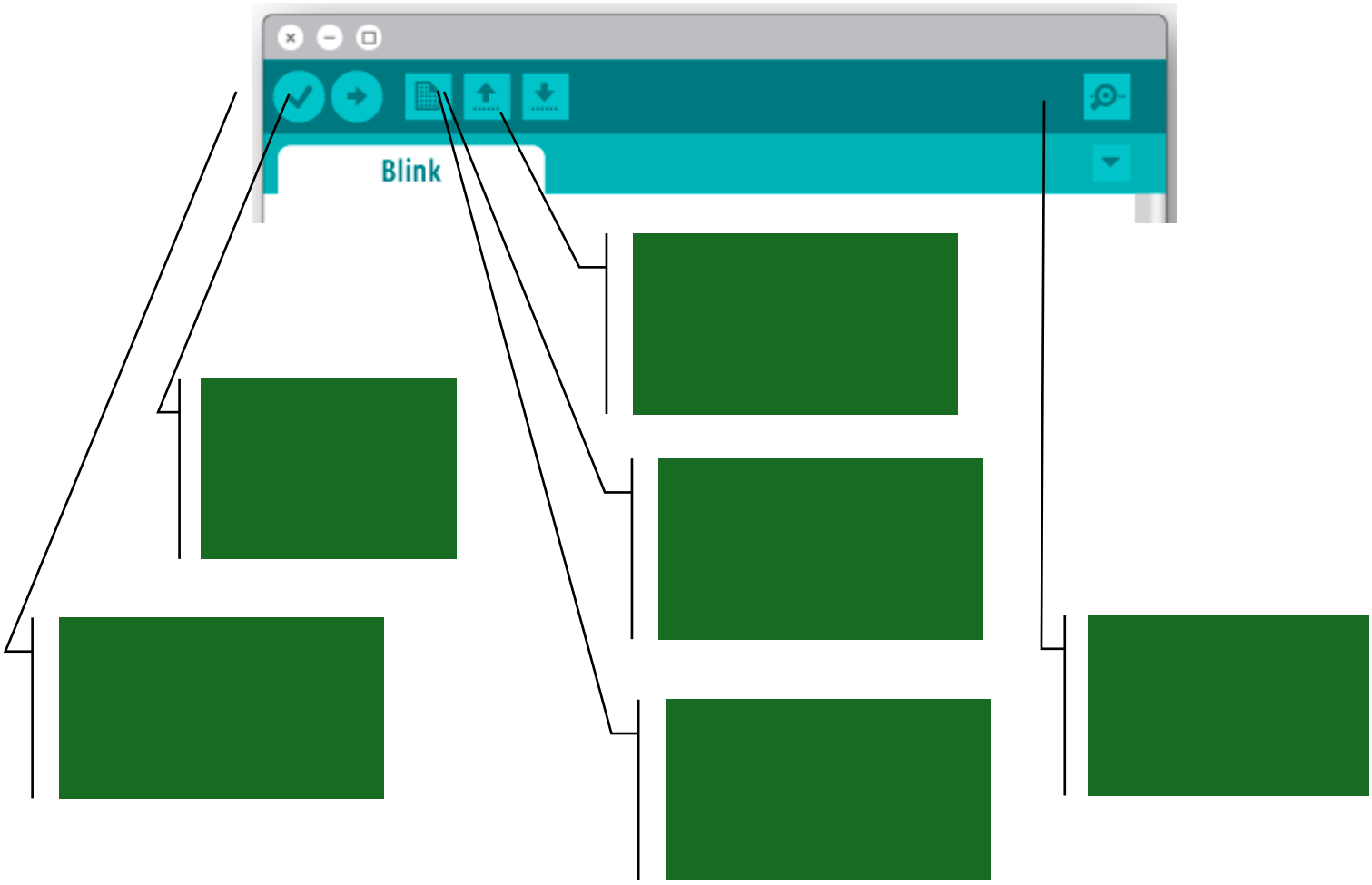
1. Go to this page to find the latest version of the Arduino IDE: **http://arduino.cc/en/Main/Software**
2. Scroll down and click on '32 bit' or '64 bit', depending on your system type, to download.
3. Go to this page and follow the tutorial for your distribution **http://playground.arduino.cc/Learning/Linux**

Task 2. Connection of board to the IDE

1. Check 'Tools → Port' and see what is available.
2. Connect the control board to your computer using USB cable.
3. Check 'Tools → Port' again and you should see a new port – that is your board.
4. Select that new port.

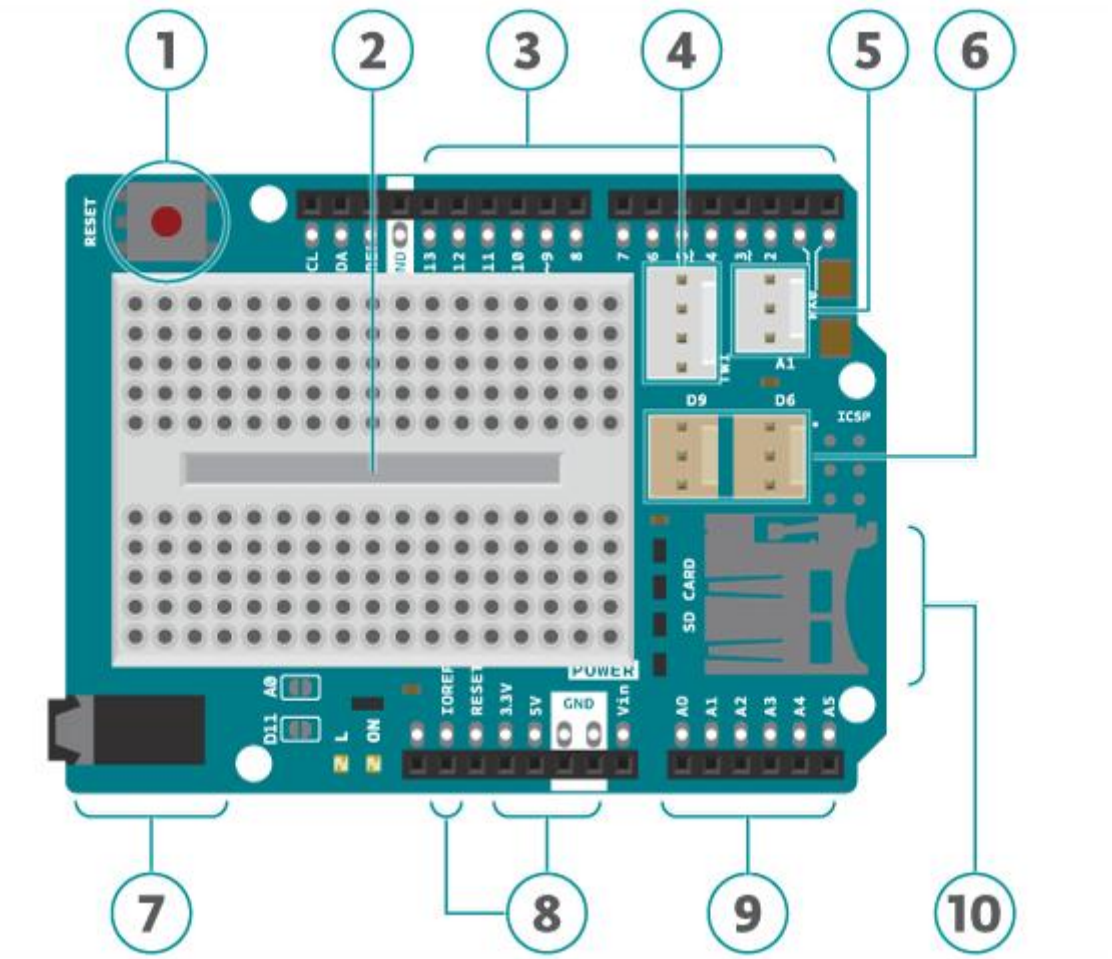
Task 3. Explanation of the features of IDE tool bar

Describe the functionality of each feature included in the IDE tool bar. Write the explanation in boxes connected to the features.



Task 4. Recognition of the control board elements

In the list below the picture, write the correct name of the board element next to its number. Use the list of elements' names: ~~reset button~~, analog input pins, I2C connector, digital 3-pin header ports, speaker plug, digital input and output pins, SD card port, breadboard, ground and power pins, A1 3-pin header port.



1. reset button

2. _____

3. _____

4. _____

5. _____

6. _____

7. _____

8. _____

9. _____

10. _____

Module 5.2. Programming of LED blinking and the sound producing using Piezo Speaker

Learning Objective:
By the end of the lesson, students will be able to understand digital signals, distinguish digital inputs and outputs, how to connect different sensors, develop simple programmes for reading external digital sensors (external LED, Speaker) and controlling digital actuators, understand and use different classes, functions and commands from the Education shield library.

Materials:
- 1 x control board
- 1 x Arduino Education Shield
- 5 x LED
- 5 x 220 ohm resistor
- 1 x piezo speaker
- 10 x jumper wire

Lesson Outline:
1. Review the concept of digital signals [1]. Show simple program how to make the on-board LED blink and explain all the code lines [2, Example 1] (5 minutes).
2. Explain students the polarity of LED. Introduce students how connect own LED on the control board [2] and to protect them from burning using resistors [3] (5 minutes).
3. Show students an example with sample program how to make external LED blink [2, Example 2.2]. Show all necessary materials (control board, LED, resistor, Arduino Education Shield, breadboard, jumper wire), explain their meaning [4, 5]. Demonstrate how to connect them to the control board (5 minutes).
3. Explain all the code lines of the sample program [2, Example 2.2. or 2.3. with the variable], the importance of the library and functions used from it (5 minutes).
3*. Explain additionally the structure of the code if students did not learn the Module 1 of this course.

- Every line of code ends with a semicolon ;.
- Blocks of code are contained within curly brackets { }.
- Functions start with a definition like void, for example `void setup(){…}`.
- Functions have parameters contained between brackets ( ), for example, `pinMode(inputPin, INPUT); digitalWrite(ledPin, HIGH);` `piezo.play(numberOfNotes, melody, noteDurations, 1);`. The amount of parameters depend on the function.

4. Main exploration (20 minutes)
4.1. Let students to fulfil the Task described in [2, Example 2.2]. Ask students to connect their own LED and other materials to the board.
4.2. Copy the code of [2, Example 2.2]. Ask students to connect a 220 ohm resistor to another digital pin on the breadboard (different than given in the Example 2.2.) using a jumper wire.
4.3. Ask students to make changes in the code [2, Example 2.2.] and use a variable, for example, `int PinNumber`, to hold the digital pin number where the LED is connected, instead of just writing the number in the function `pinMode()`. See example [2, Example 2.3.]. Compile and upload the program.

4.4. Ask students to change the times inside the delay function using the code [2, Example 2.2 or 2.3]. Test different ways to make the light blink faster or slower.
4.5. Test to see what happens when the delay becomes really small, for example `delay(10)`, and make conclusions. Ask students to fill out the Task 1, i.e. report experiment results in the table.
5. Explain the production of the sound by generating vibrations using a piezoelectric buzzer (piezo speaker), making it beep. Explain the piezo speaker device and how it works (3 minutes).
6. Introduce students how to connect piezo speaker on the control board [6] and explain all the code lines of the sample program [2, Example 2.4. or 2.5.] (5 minutes)
7. Explain the names of tones and their frequency and how to play different tones by using function `tone()` [6]. Show an code example where the `tone()` function is used to generate an A with a frequency of 440 hertz (5 minutes).
9. Exploration (7 minutes)
– Ask students to compile and upload the program [6, example 2.4. or 2.5.].
– Ask students to generate an C, D, E, F, G or B by changing the code of sample program [6, example 2.6.]. Let students to change the duration of sound by changing the third parameter of the function `tone()`.
– Ask students to fulfil the Task 2.
10. Explain how several LEDs can be connected together in a line using VU-meter and to control them using the EducationShield library. Show the example with 5 LEDs [7]. Explain operators and functions used in the sample code such as `VuMeter me`, `me.config()`, `me.begin()`, `me.fill()`,`me.clear()`, `me.on()`, `me.off()`, `me.blink()`, `me.blinkAll()`, `me.scrollLeft()`, `me.scrollRight()`, `me.fillFrom()` (8 minutes).
11. Exploration (12 minutes)
Ask students to:
– fulfil the Task 3;
– connect three LEDs on board according to the instructions given in [7];
– copy the sample code from [7 or using IDE tool bar File>Example>EducationShield>Help>vuMeter], compile and upload the program;
– make changes in the code to achieve such a result:

1. blink all LEDs 5 times for 1.5 secods;
2. turns on one LED at a time from left to right, each LED should be turned on for 2 seconds;
3. turns on one LED at a time from right to left, each LED should be turned on for 1 second;
4. blink all LEDs 5 times for 1.5 secods;
5. blinks 2nd and 3rd LED with delay 2 seconds;
6. blinks only 3rd LED with delay 2 seconds;
7. turn on all LEDs for 5 seconds;
8. turn off 1st LED for 3 second;
9. turn on 1st LED for 2 seconds;
10. turn off all LEDs.

9. Reflection (5 minutes)

– Students summarize what they learned and share personal insights or preferences regarding programming control board, reading external digital sensors (external LED, Speaker) and controlling digital actuators.

Additional tasks:

- Use the light to simulate a heartbeat, something like tick tock … pause … tick tock … pause ….
- Write a program that plays a sound when you push a button.
- Add an LED that blinks when the button is pushed.
- Write a melody using `tone()` [8].
- Write a melody using the `play()` function in the EducationShield library.

Sources:

[1] https://ctc101.arduino.cc/resources?id=32372703
[2] https://ctc101.arduino.cc/ctc101/module/module-2/lesson/blink
[3] https://ctc101.arduino.cc/resources?id=32372703
[4] https://ctc101.arduino.cc/resources?id=29415213
[5] https://ctc101.arduino.cc/resources?id=31800414
[6] https://ctc101.arduino.cc/ctc101/module/module-2/lesson/beep
[7] https://ctc101.arduino.cc/resources?id=32379590
[8] https://ctc101.arduino.cc/resources?id=32150446

Worksheet "Exploring LED"

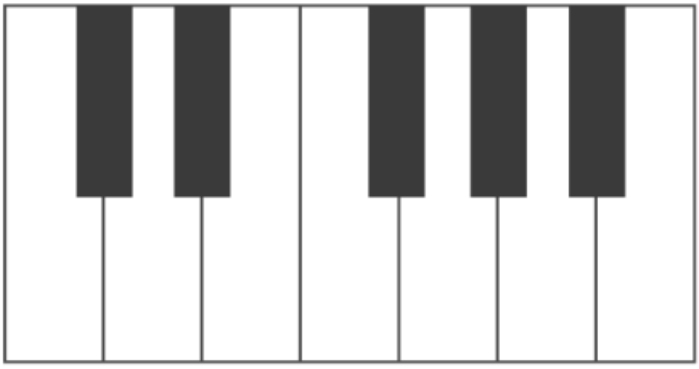
Task 1. Reporting the experiment results of the light blinking

No.	Value of the parameter in the 1st `delay()` function	Value of the parameter in the 2nd `delay()` function	Explanation of the observed result
for example	1000	1000	the LED blink with a 2 second interval
1.			
2.			
3.			
4.			
5.			
6.			
7.			
8.			
9.			
10.			

Worksheet "Exploring Speaker"

Task 2. Generation of short melody

Generate the given melody from 5 notes, for example, Do-Mi-Sol-Mi-Do. Write the code using function `tone()` and tone frequency explanation given in the Figure 2.1.



TONE	FREQUENCY (in hertz)	DELAY VALUE
Do (C)	261.63	1915 µs
Re (D)	293.66	1700 µs
Mi (E)	329.63	1519 µs
Fa (F)	349.23	1432 µs
Sol (G)	392.00	1275 µs
La (A)	440.00	1136 µs
Si (B)	493.88	1014 µs

Figure 2.1. Tone frequency explanation

For example,

This example, we use the `tone()` function to generate a melody from three notes Do-Fa-La, playing each tone for 2 seconds.

```
void setup()
{
      tone(8, 261.63, 2000);
      tone(8, 349.23, 2000);
      tone(8, 440, 2000);
}
void loop() { }
```

Worksheet "Exploring EducationShield library"

Task 3. Recognition of the functions of the EducationShield library

Connect the correct description of the function or operator with its name.

`VUMeter`	turns LEDs on from the first to length
`config(int length, int pins)`	turns all LEDs off
`begin()`	turns on LEDs from startIndex to stopIndex
`clear()`	creates a VUMeter object
`on(int index)`	stops the program for a certain amount of time
`scrollRight( int speed,int startIndex)`	blinks all LEDs
`blink(int index, int speed,int times)`	initializes the the VU-meter
`off(int index)`	turns on one LED at a time from left to right
`fillFrom(int startIndex, int stopIndex)`	Allows to configure specific PIN as an output using function pinMode().
`fill(int length)`	configures the VUmeter
`setup()`	repeats given block of code
`digitalWrite(pinNumber, HIGH/LOW)`	turns one LED on.
`blinkAll(int speed,int times )`	blinks one LED
`delay(int time )`	turns one LED off
`loop()`	writes HIGH or LOW to the digital pin pinNumber

Module 5.3. Project development: building and playing small electronic game that simulates sport games

Learning Objective:
By the end of the lesson, students will be able to understand how to develop own project by tracking the score using a light sensor, phototransistor, LEDs, piezo speaker or another sensor, summarize and present project results, work individually and in group.

Materials:

1. Project "Bascetball"

- 1 x control board
- 1 x Arduino Education Shield
- 1 x light sensor module
- 1 x piezo speaker
- 5 x LED
- 5 x 220 ohm resistor
- 1 x module cable
- 12 x jumper wire

Other materials:

- 1 x ping-pong ball
- 1 x tape
- 1 x scissor
- 1 x plastic cup (several may be needed if some break)

2. Project "React"

- 1 x control board Arduino 101
- 1 x Arduino Education Shield
- 1 x breadboard
- 1 x piezo speaker
- 3 x LED
- 3 x 220 ohm resistor
- 3 x 1 Mohm resistor
- 18 x jumper wire (3 long)

Other materials:

- 3 x aluminum foil piece
- 1 x masking tape roll

Lesson Outline:

1. Repeat one of the projects using instructions given in [1 or 2], working individually or in pairs (15 minutes).
2. Students make changes in the sample project, i.e. come up with a different variation of the chosen game (25 minutes). Some ideas for changes are provided below.

Project "Bascetball"

- add more LEDs and increase the possible score
- make up own melody
- change the way the LEDs blink when it is game over

Project "React"

- change the reaction time and the waiting time
- make up own melody
- change the way the LEDs blink when it is game over
- try adding more LEDs and sensors

3. Students demonstrate their projects and explain changes made (40 minutes).
4. Students develop their own project, describe project and its result using Template 1. Project description, working individually or in pairs. Let students select any components (sensors etc.) from the Arduino CTC 101 set, generate idea of their own project and implement it (50 minutes).
5. Students present their projects (30 minutes).

Worksheet "Project template"

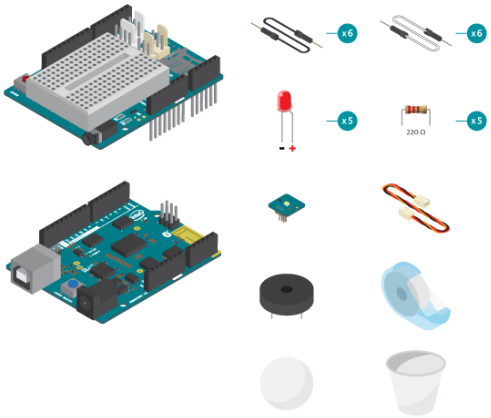
Project name	**Title of the project**
Project author	**Name Surname of student**
Date of creation	DD.MM.YYYY
Date of review	DD.MM.YYYY
Mark	**Points or %**

DESCRIPTION OF THE PROJECT IDEA

For example,
In this game, players will try to land a ping pong ball into a cup. Make five points to win. The score is tracked using a light sensor, or more specifically a phototransistor.

MATERIALS

Note: additionally, student can add pictures of components.



- 1 x control board
- 1 x Arduino Education Shield
- 1 x light sensor module
- 1 x piezo speaker
- 5 x LED
- 5 x 220 ohm resistor
- 1 x module cable
- 12 x jumper wire

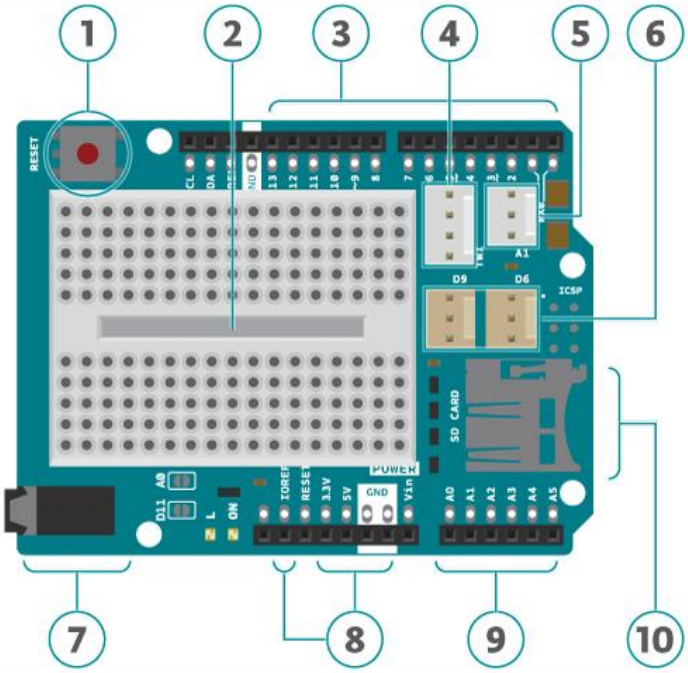
Other materials:

- 1 x ping-pong ball
- 1 x tape
- 1 x scissor
- 1 x plastic cup (several may be needed if some break)

INSTRUCTIONS

1. Attach the shield onto the top of the board.

2. Connect five LEDs across the breadboard gap.
3. Connect a 220 ohm resistor to digital pin 2. Then connect the resistor to the long leg of the first LED.
4. Connect each of the digital pins 3 through 6 to a corresponding LED following the same method.
5. Connect the short legs of the LEDs to GND.
6. Connect the piezo speaker across the breadboard gap and connect one leg to digital pin 8 (it does not matter which one), then the other one to GND.
7. Connect the light sensor module to the analog module connector, A1 on the shield.
8. Use the module cable to connect the module to the shield.
9. Get some tape, scissors and a plastic cup.
10. Cut a hole in the bottom of the plastic cup so the light sensor fits.
11. Place the light sensor in the hole.
12. Tape the sensor to the cup.
13. Make sure that the sensor is exposed through the hole in the cup (see Scheme).



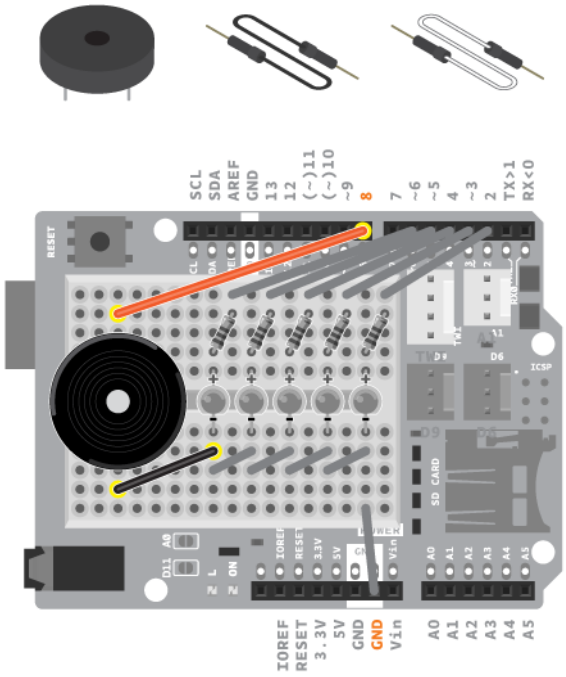
Scheme template

Note: Students should draw components on the board, where and how they should be connected, using coloured markers or pencils.

14. Check that your wiring is ready and connect the board to the computer.
15. Find the example LightSensorTest program and open it. You can use this program to calibrate the light sensor and set the threshold value.
16. Upload the program to the board.
17. Go to the Serial communication reference and learn how it works. Then open the Serial Monitor.

18. Check the sensor value when the ball is not in the cub, and write it down.
19. Place a ping-pong ball in the cup, over the light sensor. Write down the covered value. Average both numbers (uncovered and covered) for threshold value.
20. Now that you know the uncovered and average, check that your wiring is ready and connected to the computer.
21. Find the Basketball program and open it.

22. Find the syntax line sensor.config(uncovered , threshold), changing the values in the parameters to the uncovered value and threshold value respectively. Then upload the program to the board.
23. Start to play basketball.



Scheme: Example of Scheme

or

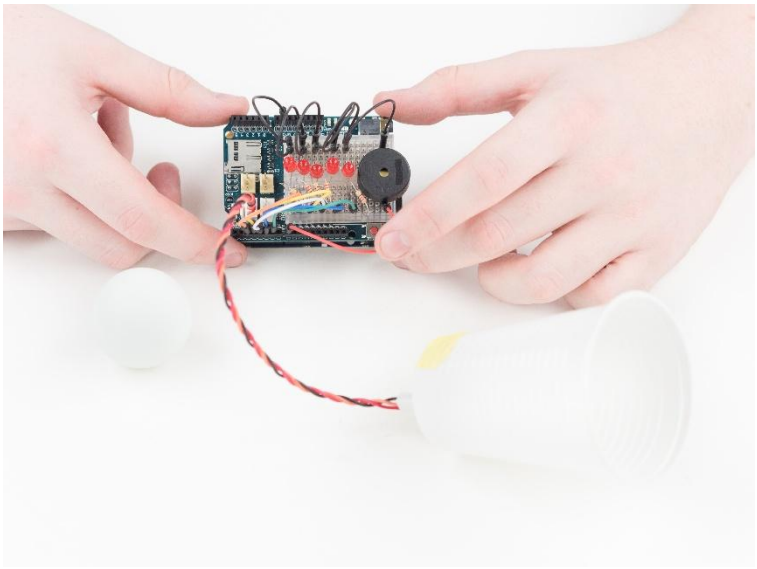


Figure: Example picture of the board

CODE

```
Insert the program code
```

Module 6. Basics of Analog Signals and Serial Communication

Aim: Understand the reading and writing analog signals. Learn the basics of serial communication to connect the control board to the computer and exchange data with the computer.

Outline:

Module 3.1. Reading and writing analog signals, and usage of analog sensors to program robots.
Module 3.2. Serial communication between the control board and computer for sending and receiving data
Module 3.3. Project development: building an element that simulates some process (generation of music, display message, play beats)

Time/hours: 6 (3 double lessons - 2x40 min)

Module 6.1. Reading and writing analog signals, and usage of analog sensors to program robots

Learning Objective:
By the end of the lesson, students will be able to understand analog signals and create simple projects, read and write analog signals, to measure some parameters applying different sensors.

Materials:
- the 101 control board
- 1 x Education Shield
- 1 x potentiometer
- 3 x jumper wire
- 1 x LED
- 1 x 220 ohm resistor
- Projector or board for teacher demonstration

Lesson Outline:
1. Explain students the meaning of analog signals, examples of analog sensors (such as sound sensors, light sensors, temperature sensors, humidity sensor, pressure sensors, proximity, colour, gas, radiation sensors etc.) and real-world examples of analog sensor usage to measure some things [1, 5] (5 minutes).
2. Explain students the potentiometer [2] and its applications such as the volume controller, dimmer switches for lights, brightness controls in televisions, and faders in audio equipment (5 minutes).
3. Explain students what components are necessary for the practical part [1, Example 3.1.] (2 minutes).
4. Provide students the instructions how to connect potentiometer to the control board and which commands to use for signal reading. Explain each code line of the [1, Example 3.1.] (3 minutes).
5. Ask students to fulfil Example 3.1. following instructions (15 minutes).

6. Explain students how to write analog signals. Explain students what components are necessary for the practical part [3, Example 3.2.] and code of the example (5 minutes).
7. Ask students to fulfil Example 3.2. following instructions (15 minutes).
8. Ask students to make changes in the code with the aim to (15 minutes):
- change the speed of "breathing" so that it fades faster/slower;
- add potentiometer and apply it to control the LED;
- fulfil Examples 3.3. and 3.4. [4].
8. Reflection (10 minutes)
– Ask students answer questions:

- Mention real world examples of analog signals.
- Mention examples of analog sensors.
- What pins use functions analogread() and analogwrite()?
- What values (min, max) read function analogread()?
- What values (min, max) read function analogwrite()?

– Students summarize what they learned and share personal insights or preferences regarding analog sensors and applications of potentiometers.

Sources:
[1] https://ctc101-dev.arduino.cc/ctc101/module/module-3/lesson/reading-analog-signals
[2] https://ctc101-dev.arduino.cc/resources?id=32151800
[3] https://ctc101-dev.arduino.cc/ctc101/module/module-3/lesson/writing-analog-signals
[4] https://ctc101-dev.arduino.cc/ctc101/module/module-3/lesson/light-sensor
[5] https://components101.com/article/different-types-of-sensors-and-sensing-technologies

Module 6.2. Serial communication between the control board and computer for sending and receiving data

Learning Objective:
By the end of the lesson, students will be able to understand serial communication, how to connect control board to the computer, send data from control board to the computer and receive data from the computer.

Materials:
1 x Education Shield
1 x USB cable
1 x light sensor module
1 x module cable
Projector or board for teacher demonstration

Lesson Outline:
1. Explain students how control board communicates with the computer and show how to connect it to the computer [1]. Explain and show pins on the control board which are used to send and receive data from/to control board [2, Example 3.5.] (5 minutes).
2. Demonstrate students an example of how to send text message to the serial port [2, Example 3.5.] and explain the code. Change the text message. (7 minutes)
3. Connect the light sensor module. Demonstrate students an example with sample program how to read values of light sensor [2, Example 3.6]. Explain each code line (8 minutes).
4. Ask students to fulfil Task1 by following instructions [2, Example 3.6]. Report results to the table. (20 minutes)
5. Explain commands used for receiving data over the serial port. Demonstrate an example [3, Example 3.7.] by explaining all the code lines (10 minutes).
6. Ask students to check any of the examples in File -> Examples -> EducationShield-> Help where the serial port is used. (35 minutes)
9. Reflection (5 minutes)
– Ask students to answer questions:

- Which data can be sent or received through serial port?
- How many and what pins serial port uses?
- Which commands are used to print the text or values?

– Students summarize what they learned and share personal insights or preferences regarding programming control board, reading external digital sensors (external LED, Speaker) and controlling digital actuators.

Sources:
[1] https://ctc101-dev.arduino.cc/ctc101/module/module-3/lesson/serial-port
[2] https://ctc101-dev.arduino.cc/ctc101/module/module-3/lesson/sending-to-computer
[3] https://ctc101-dev.arduino.cc/ctc101/module/module-3/lesson/receiving-from-computer

Worksheet "Exploring Serial Communication"

Task 1. Measure the light intensity

1. Create project by following instructions [1, Example 3.6.]
2. Declare a variable that holds a mapped value of the sensor reading (use the function map()).
3. Print both values to the serial monitor and see how they change when you interact with the light sensor.
4. Make experiments and measure the light intensity in different places/rooms/sides of the room.
5. Report results to the table.

Date, time	Place	Light sensor value

Module 6.3. Project development: building an element that simulates some process (generation of music, display message, play beats)

Learning Objective:
By the end of the lesson, students will be able to understand how to develop the project following instructions using different sensors (for example, light sensor, phototransistor, LEDs, piezo speaker) and other components (for example, button, web camera) following instructions, work individually and in group, present project results.

Materials:

1. Project "Binary LP"
- 1 x control board Arduino 101
- 1 x Arduino Education Shield
- 1 x IR Array
- 1 x piezo speaker
- 7 x jumper wire (5 long)
- 1 x Binary LP kit

Other materials:
- 1 x Binary LP paper disc
- 1 x tape
- 1 x scissor
- 1 x printer

2. Project "Boombox"
- 1 x control board
- 1 x Arduino Education Shield
- 1 x micro SD card
- 3 x button
- 1 x speaker
- 3 x 10 Kohm resistor
- 9 x jumper wires

3. Project "Cookie monster"
- 1 x control board Arduino 101
- 1 x Arduino Education Shield
- 1 x breadboard
- 1 x web camera
- 1 x power LED module
- 1 x 1 Mohm resistor
- 1 x module cable
- 3 x jumper wire (1 long)
- 1 x Cookie Monster kit

Other materials:
- 1 x Aluminum foil piece
- 1 x conductive cookie jar
- 3-4 x rubber band
- 1 x tape

4. Project "Drawdio"
- 1 x control board Arduino 101
- 1 x Arduino Education Shield
- 1 x breadboard
- 1 x piezo speaker
- 1 x 1 Mohm resistor
- 5 x jumper wire

Other materials:
- 1 x graphite pencil
- 1 x tape

5. Project "Cnock cnock box"
- 1 x control board Arduino 101
- 1 x Arduino Education Shield
- 1 x breadboard
- 2 x piezo speaker

6. Project "P.O.V"
- 1 x control board
- 1 x Arduino Education Shield
- 5 x LED
- 5 x 220 ohm resistor

- 1 x 1 Mohm resistor
- 1 x 9V battery
- 1 x power plug
- 1 x 3v3 zener diode
- 6 x jumper wire
- 1 x Knock Knock box kit

Other materials:
- 1 x tape

- 10 x jumper wire
- 1 x 9V battery
- 1 x power plug

7. Project "Sequencer"
- 1 x control board
- 1 x Arduino Education Shield
- 1 x micro SD card
- 1 x 8 ohm speaker
- 5 x 680 ohm resistor
- 1 x 220 ohm resistor
- 1 x 470 ohm resistor
- 1 x 1.2 Kohm resistor
- 9 x jumper wire
- 9 x loose wire
- 1 x Sequencer kit

Lesson Outline:

6. Repeat one of the projects using provided kits and instructions given in [1 - 7], working individually, in pairs or bigger group (30 minutes).
7. Students make changes in the code of sample project (25 minutes).
8. Students demonstrate their projects and explain changes made (25 minutes).

Sources:
[1] https://ctc101-dev.arduino.cc/ctc101/module/module-3/project/binary-lp
[2] https://ctc101-dev.arduino.cc/ctc101/module/module-3/project/boombox
[3] https://ctc101-dev.arduino.cc/ctc101/module/module-3/project/cookie-monster
[4] https://ctc101-dev.arduino.cc/ctc101/module/module-3/project/drawdio
[5] https://ctc101-dev.arduino.cc/ctc101/module/module-3/project/knock-knock-box
[6] https://ctc101-dev.arduino.cc/ctc101/module/module-3/project/pov
[7] https://ctc101-dev.arduino.cc/ctc101/module/module-3/project/sequencer

Module 7. Working with motors

Aim: Understand and apply the principles of using motors with Arduino CTC 101, including standard and continuous servo control.

Outline:
Explore different types of motors and their functions in mechatronic systems.
Learn how to connect, control, and program standard and continuous servo motors.
Create input-controlled motion using sensors and variables.
Design and build a small robot or motion-based project.
Apply creativity and teamwork in a final challenge or competition involving multiple servos.

Time/hours: 6 (3 double lessons - 2x40 min).

Module 7.1. Types of Motors and Standard Servo

Aim: Understand the basic types of motors and learn how to connect and control a standard servo motor using Arduino.

Learning Objective:
By the end of the session, students will be able to:

- Identify and describe the main types of motors (DC, servo, stepper).
- Explain how a standard servo motor works.
- Connect and control a standard servo motor using Arduino code.
- Observe how servo position corresponds to code values.

Materials:
Arduino board and USB cable
Servo motor (standard)
Jumper wires
Breadboard
Laptop with Arduino IDE installed
Worksheet “Exploring Servo Motors”
Projector or board for teacher demonstration

Lesson Outline:

1. The teacher explains different types of motors (DC, servo, stepper) with images or short videos. Discussion: Where do we see motors in everyday life? [1] (15 minutes)
2. The teacher explains what makes a servo different: controlled by position rather than speed. Introduces key terms: pulse width modulation (PWM), angle range (0–180°). [2] (10 minutes)

3. The teacher demonstrates how to connect the servo motor to Arduino (Signal–Power–Ground) [2]. Students repeat the setup in pairs. (10 minutes)
4. The teacher shows example code using the Servo library [2][3]: (15 minutes)

```
#include <Servo.h>
Servo myServo;

void setup() {
  myServo.attach(9);
}

void loop() {
  myServo.write(0);
  delay(1000);
  myServo.write(90);
  delay(1000);
  myServo.write(180);
  delay(1000);
}
```

Discuss what happens and why the servo pauses.

5. Students modify the code to create their own movement pattern. The teacher encourages creativity and checks connections. (20 minutes)
6. Students answer reflection questions on their worksheet “Exploring Servo Motors”. (10 minutes)

Reflection (5 minutes):
Before leaving the class, each student answers the question: What are two real-world uses of servo motors?

Extension Ideas:

- Create a servo that moves smoothly from one angle to another.
- Add LED indicators showing the servo’s current direction.
- Research where stepper motors are used and how they differ from servos.

Sources:
[1] https://ctc101.arduino.cc/ctc101/module/module-4/lesson/types-of-motors
[2] https://ctc101.arduino.cc/ctc101/module/module-4/lesson/standard-servo
[3] https://ctc101.arduino.cc/ctc101/module/module-4/lesson/continuous-rotation-servo

Worksheet "Exploring Servo Motors"

Learn how to connect and control a standard servo motor using Arduino.

Task 1. Identifying Motor Types

Name three common types of motors and their main use:

Type of Motor	Example Use
1	
2	
3	

Task 2. Connecting the Servo

Learn how to correctly connect a standard servo motor to the Arduino board.

Instruction:

1. Take one standard servo motor from your kit.
2. Identify its three wires:
   - Orange / Yellow -> Signal
   - Red -> Power (+5V)
   - Brown / Black -> Ground (GND)
3. Use jumper wires to connect the servo to your Arduino board:

Servo Wire	Connects to on Arduino	Description
Signal (orange/yellow)	Pin 9	Sends control signal
Power (red)	5V	Provides power
Ground (brown/black)	GND	Completes the circuit

Draw your circuit below, label all connections clearly (Pin 9, 5V, Ground).:

If you have coloured pencils, use the correct colours for wires.

Task 3. Controlling the Servo

Study the example code and write your own version below.

```
#include <Servo.h>
Servo myServo;
```

```
void setup() {
  myServo.attach(9);
}

void loop() {
  myServo.write(0);
  delay(1000);
  myServo.write(90);
  delay(1000);
  myServo.write(180);
  delay(1000);
}
```

Modify the code by changing positions or timing.

Task 4. Observation
Study the example code and write your own version below.
Run your code.
Describe what happens:

Module 7.2. Continuous Rotation and Input Control

Aim: Use sensors and servo motors together to create an interactive mini-project: a robotic arm that waves or reacts to user input.

Learning Objective:
By the end of the session, students will be able to:

- Connect and control a servo motor using input signals from a potentiometer or light sensor.
- Understand how analog input values influence motor behaviour.
- Combine programming, sensors, and actuators into a small, functioning project.
- Practice structured coding and debugging.

Materials:
Arduino CTC 101 Kit
Jumper wires and breadboards
Computers with Processing IDE and Python mode installed
Worksheet “Input-Controlled Servo Challenge”
Projector or board for teacher demonstration

Lesson Outline:

1. The teacher introduces the concept of interactive systems, robots that respond to their environment. Examples: automatic doors, robotic pets, smart toys.[1][2][3][4][5] (5 minutes)
2. The teacher shows a prebuilt example- a “waving arm” made of a servo attached to a cardboard cutout, moving based on potentiometer input. Explains that the servo position (0°–180°) changes with the input sensor value (0–1023). [6][7] (10 minutes)
3. Students connect a potentiometer to A0 and a servo to Pin 9 according to the teacher’s example. They verify the wiring using the Worksheet “Input-Controlled Servo Challenge”. (15 minutes).
4. The teacher guides students through writing code to read the potentiometer and control the servo. Students type and test this example:

```
from Arduino import Arduino
import time

board = Arduino("9600")

pot_pin = 0   # Analog input A0
servo_pin = 9 # Digital pin 9

board.pinMode(servo_pin, "SERVO")

while True:
    val = board.analogRead(pot_pin)
    angle = int((val / 1023) * 180)
    board.analogWrite(servo_pin, angle)
    time.sleep(0.05)
```

The teacher explains how the analog input is mapped to a servo angle (0–180°). (20 minutes)

5. Students decorate their servo with simple cardboard arms or figures to make it wave, nod, or turn. Encourage creativity (for example, waving robot, cat tail, moving flag). (15 minutes)
6. Volunteers demonstrate their interactive robots. The teacher leads discussion: What changes could make the motion smoother or more reactive? (10 minutes)

Reflection (5 minutes):
Students write one short sentence about what they found most challenging: wiring, coding, or mapping values, and explain why.

Extension Ideas:

- Replace the potentiometer with a light sensor to make a “light-reactive waving robot.”
- Add two servos for two arms that wave alternately.
- Control the servo with Processing interface sliders on-screen (for advanced students).

Sources:
[1] https://ctc101.arduino.cc/ctc101/module/module-4/project/camera-robot
[2] https://ctc101.arduino.cc/ctc101/module/module-4/project/light-chaser
[3] https://ctc101.arduino.cc/ctc101/module/module-4/project/line-follower
[4] https://ctc101.arduino.cc/ctc101/module/module-4/project/magic-box
[5] https://ctc101.arduino.cc/ctc101/module/module-4/project/tickle-robot
[6] https://ctc101.arduino.cc/ctc101/module/module-4/lesson/input-controlled-servo
[7] https://www.instructables.com/Waving-Arm/

Worksheet "Input-Controlled Servo Challenge"

Build a simple interactive servo project that responds to user input (e.g., a potentiometer) to make a robot arm wave or move.

Task 1. Build the Circuit

Connect a servo motor and potentiometer to your Arduino correctly.

Instructions:

1. Use one standard servo and one potentiometer from your Arduino CTC 101 kit.
2. Make the following connections:

Component	Pin on Arduino	Description
Servo- Signal (Orange/Yellow)	Digital 9	Control signal
Servo- Power (Red)	5 V	Power supply
Servo- Ground (Brown/Black)	GND	Common ground
Potentiometer- Middle Pin	Analog A0	Reads analog value
Potentiometer- Side Pin 1	5 V	Voltage input
Potentiometer- Side Pin 2	GROUND	Ground

Draw your circuit below and label all wires clearly:

Check:

☐All servo and potentiometer wires are connected properly.

☐Power and ground are correct.

☐Servo connected to PWM pin (9 or 10).

Task 2. Write the Code

Make the servo move according to the potentiometer's position.

Code Example:

```
from Arduino import Arduino
import time

board = Arduino("9600")

pot_pin = 0     # Analog A0
servo_pin = 9   # Digital 9

board.pinMode(servo_pin, "SERVO")
```

```
while True:
    val = board.analogRead(pot_pin)     # Read potentiometer (0–1023)
    angle = int((val / 1023) * 180)     # Convert to 0–180 degrees
    board.analogWrite(servo_pin, angle) # Move servo
    time.sleep(0.05)
```

Write your edited or customized code version.

Task 3. Customize Your Robot Arm
Give your servo some personality. Ideas:

- Attach a cardboard arm or flag to your servo horn.
- Decorate it as a robot, cat tail, or waving hand.
- Use markers, paper, or tape. Be creative!

Draw or describe your creation below:

_____

_____

_____

Task 4. Observe and Reflect
Test your setup:

1. Turn the potentiometer slowly.
2. Watch the servo's movement.

Write the answer. What happens when the potentiometer is at minimum / maximum position?

_____

_____

Module 7.3. Crawling Creatures Competition

Aim: Design, build, and program a crawling robot creature using servo motors. Work in teams to test, improve, and compete to see whose creature crawls the farthest or moves in the most creative way.

Learning Objective:
By the end of the session, students will be able to:

- Apply their knowledge of servo motors to create movement in a robot.
- Experiment with mechanical design and materials to produce crawling motion.
- Write and test Arduino code to control motion timing and patterns.
- Collaborate effectively in small teams to plan, build, and present a working prototype.
- Evaluate and refine their designs through testing and competition.

Materials:
Arduino CTC 101 Kit
Jumper wires and breadboards
Computers with Processing IDE and Python mode installed
Cardboard, straws, paper clips, rubber bands, tape, etc. (for the robot's body and limbs)
Worksheet "Crawling Creatures Challenge"
Measuring tape or ruler (for competition)
Timer or stopwatch (can be a phone timer or physical stopwatch)

Lesson Outline:

1. The teacher shows short clips or images of simple crawling robots and biological inspirations (e.g., insects, worms, crabs). [1][2][3] The teacher explains the challenge: Each team will design and build a creature that can move forward using at least one servo motor. At the end of class, all creatures will compete for distance and creativity awards. (10 minutes)
2. Students form teams (2–3 people) and brainstorm:
    - What will your creature look like?
    - How will it move? (sliding, wiggling, stepping, hopping)
    - What materials will you use?
    - Where will you attach the servo(s)?

Teams sketch their design and list materials on the worksheet "Crawling Creatures Challenge" (Task 1). (10 minutes)

3. Teams build their robot and write the code:
    - Assemble their creature from available materials.
    - Connect the servo(s) to the Arduino.
    - Upload and test motion code.

Example Code:

```
#include <Servo.h>
Servo servo1;
Servo servo2; // optional second servo

void setup() {
```

```
  servo1.attach(9);
  servo2.attach(10); // optional
}

void loop() {
  servo1.write(60);
  delay(500);
  servo1.write(120);
  delay(500);

  servo2.write(120);
  delay(500);
  servo2.write(60);
  delay(500);
}
```

Teams complete Task 2 and Task 3 on the worksheet “Crawling Creatures Challenge”. (40 minutes)

4. Teams test their robots on a flat surface. Students record results, identify what works, and make improvements to movement, balance, or structure; complete Task 4 on the worksheet “Crawling Creatures Challenge”. (10 minutes)
5. Each team presents their robot for the final challenge. (15 minutes)

Competition Categories:

- Distance Champion: travels the longest forward distance in 10 seconds.
- Smoothest Motion: most realistic or stable movement.
- Best Design: most creative creature concept and decoration.

The teacher or judges measure distance and record their results. Teams fill Task 5 on the worksheet “Crawling Creatures Challenge”.

Reflection (5 minutes):
Students write about one challenge they faced while making the robot and how they solved it.

Extension Ideas:

- Add a sensor (light or sound) to make the creature react to its environment.
- Use two servos for coordinated limb movement.
- Create a Zoo of creatures for an exhibition or race event.
- Organize a time-based race: measure how long it takes for each robot to crawl 50 cm in a straight line. The robot that reaches the finish line fastest wins the Speed Champion title. Encourage teams to improve motion efficiency, stability, and direction control.

Sources:

[1] https://ctc101.arduino.cc/ctc101/module/module-4/project/crawling-robot
[2] https://www.youtube.com/watch?v=uOh2ZY1VHGE
[3] https://www.youtube.com/watch?v=S4eQXXxUnNE

Worksheet "Crawling Creatures Challenge"

Task 1. Plan Your Creature
Brainstorm and plan your project. Sketch your idea and answer:

- What kind of creature are you building?
- How will it move?
- Which parts will move using servo(s)?

Draw and label your components below:

Task 2. List Components

Component	Quantity	Function
Servo		
Wires		
Breadboard		

Task 3. Build and Program
Use your knowledge from the previous lessons to connect the servos and create code for motion. Write your motion code:

Task 4. Test and Improve
After testing:

- What worked well? _____
- What didn't work? _____
- What did you change to improve your robot? _____

_____

Task 5. Competition Results

Test run: Distance in 10 seconds = _____ cm

☐Moved forward;

☐Turned in circles;

☐Fell over;

☐Other _____

Answer the question: What would make your creature faster or more stable next time?

_____

_____

_____