WhiteHat Jr
Live online Coding for Kids

Topic	ROBOTIC ARM 3	
Class Description	**Students will learn how to attach a distance sensor with the robotic arm.**	
Class	**PRO C290**	
Class time	**50 mins**	
Goal	● Add a distance sensor to the robotic arm. ● Getting values from the distance sensor.	
Resources Required	● Teacher Resources: ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen ○ Smartphone ● Student Resources: ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen	
Class structure	**Warm-Up** **Teacher -Led-Activity 1** **Student-Led Activity 1** **Wrap-Up**	**5 mins** **20 mins** **20 mins** **5 mins**
Credit & Permissions:	**This project uses Webots, an open-source mobile robot simulation software developed by Cyberbotics Ltd.** **License**	
WARM-UP SESSION - 10 mins		
Teacher Action		**Student Action**

<table>
<tr><td>Hey &lt;student's name>. How are you? It’s great to see you! Are you excited to learn something new today?

Following are the WARM-UP session deliverables:
● Greet the student.
● Revision of previous class activities.
● Quizzes.</td><td>**ESR**: Hi, thanks!
Yes I am excited about it!

Click on the slide show tab and present the slides</td></tr>
<tr><td colspan="2">**WARM-UP QUIZ**
Click on In-Class Quiz</td></tr>
<tr><td colspan="2">**Activity Details**

Following are the session deliverables:
● Appreciate the student.
● Narrate the story by using hand gestures and voice modulation methods to bring in more interest in students.</td></tr>
<tr><td colspan="2">**TEACHER-LED ACTIVITY - 15 mins**</td></tr>
<tr><td colspan="2">**Teacher Initiates Screen Share**</td></tr>
<tr><td colspan="2">**ACTIVITY**
● **Attaching a distance sensor with the gripper hand.**</td></tr>
<tr><td>**Teacher Action**</td><td>**Student Action**</td></tr>
<tr><td>Do you remember what we did in the last class?

Great, if you have any doubts from the last class, please ask.

Note : Teacher will clear the doubts, if students have any.

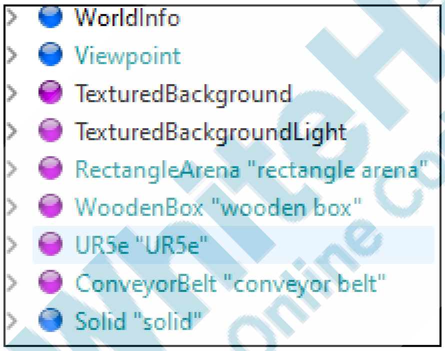
If there are no questions from the previous class, let’s jump to today's class.</td><td>**ESR :** Yes, we attached a gripper hand with the UR5e robotic arm.</td></tr>
</table>

In the last class, we faced a challenge where the gripper hand attached with the **UR5e** bot, was not able to **grip** the objects properly, as it didn't know when to claw the object.

So we discussed and came to a solution, where we decided to attach a **distance sensor** with the **gripper hand**, so that the arm can detect the objects in front of it and eventually, grab it.

To do so, let's first open the teacher boilerplate link and download all the files from it. Open all the downloaded files in the webots software.

The **scene tree** will look as shown in the graphic.

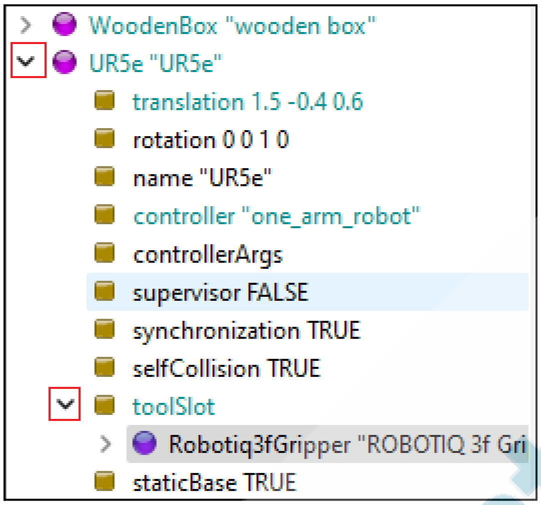
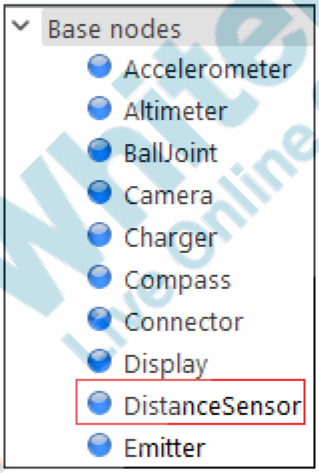


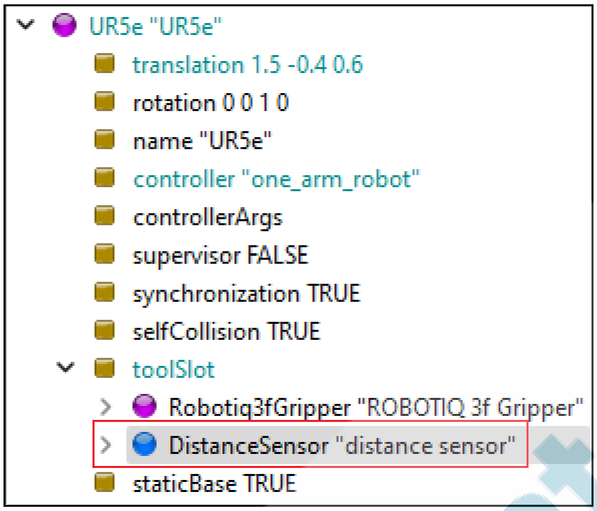
- WorldInfo
- Viewpoint
- TexturedBackground
- TexturedBackgroundLight
- RectangleArena "rectangle arena"
- WoodenBox "wooden box"
- UR5e "UR5e"
- ConveyorBelt "conveyor belt"
- Solid "solid"

Now, let's attach a **distance sensor** with the **gripper hand**, so that it's able to detect the object in front of it. For that,

a) Expand the **UR5e** node.
b) Expand the **toolslot** property.
c) Click on the **robotiq 3f gripper** '**once'** so that it gets selected.
d) Click on the **add object** button.

Note : We have selected or clicked on the gripper node 'once' before clicking on the add object button, so that the distance sensor node can come right under it.

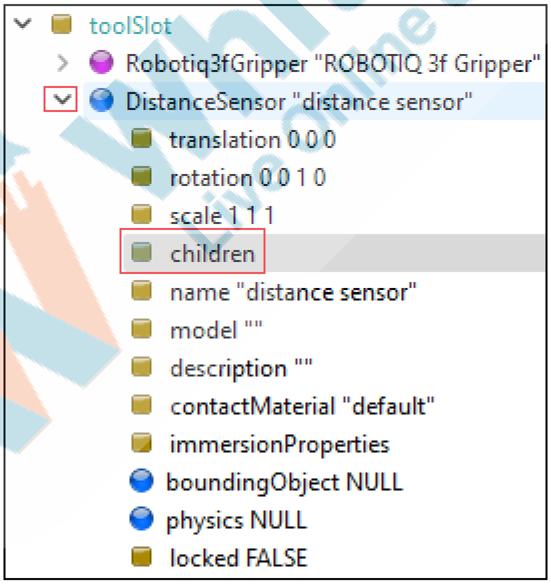
 WoodenBox "wooden box" UR5e "UR5e" translation 1.5 -0.4 0.6 rotation 0 0 1 0 name "UR5e" controller "one_arm_robot" controllerArgs supervisor FALSE synchronization TRUE selfCollision TRUE toolSlot Robotiq3fGripper "ROBOTIQ 3f Gri staticBase TRUE	
Expand **base nodes** and **double click** on the **distance sensor node**.	
 Base nodes Accelerometer Altimeter BallJoint Camera Charger Compass Connector Display DistanceSensor Emitter	
You will see that the **distance sensor node** is added under the **toolslot** property of the **UR5e** robotic arm.	



```
UR5e "UR5e"
    translation 1.5 -0.4 0.6
    rotation 0 0 1 0
    name "UR5e"
    controller "one_arm_robot"
    controllerArgs
    supervisor FALSE
    synchronization TRUE
    selfCollision TRUE
    toolSlot
        Robotiq3fGripper "ROBOTIQ 3f Gripper"
        DistanceSensor "distance sensor"
    staticBase TRUE
```

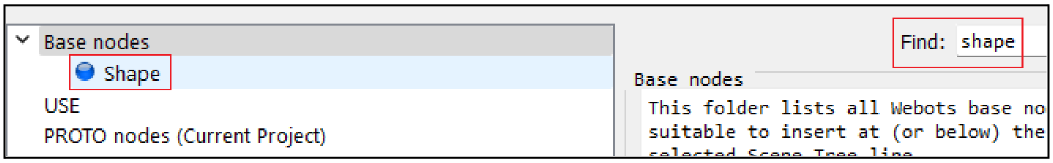
We need to add some **shape** and **appearance** to our **distance sensor node**. For that,

a) **Expand** the **distance sensor node.**
b) **Double click** on the **children** property.



```
toolSlot
    Robotiq3fGripper "ROBOTIQ 3f Gripper"
    DistanceSensor "distance sensor"
        translation 0 0 0
        rotation 0 0 1 0
        scale 1 1 1
        children
        name "distance sensor"
        model ""
        description ""
        contactMaterial "default"
        immersionProperties
        boundingObject NULL
        physics NULL
        locked FALSE
```

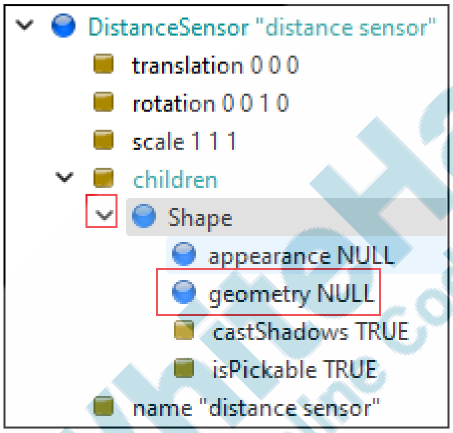
A window will appear. **Search** for the word **'shape'** in the **Find textbox** and once the search results are loaded, **double click** on the **shape** node to add it.



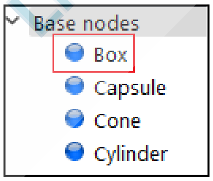
This folder lists all Webots base no
suitable to insert at (or below) the
selected Scene Tree line

Let’s give a **proper shape** to our distance sensor node. For that,

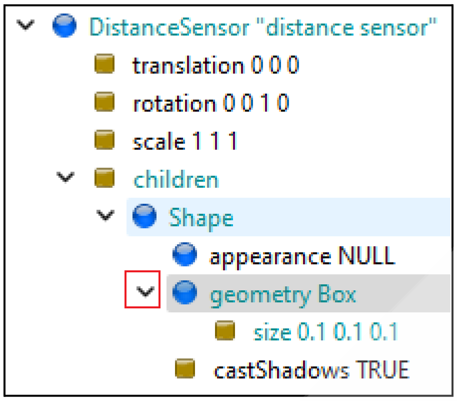
a) **Expand** the **shape** node.
b) **Double click** on the **geometry** property.



A window will appear. Expand the **base nodes** and double click on the **box** shape.

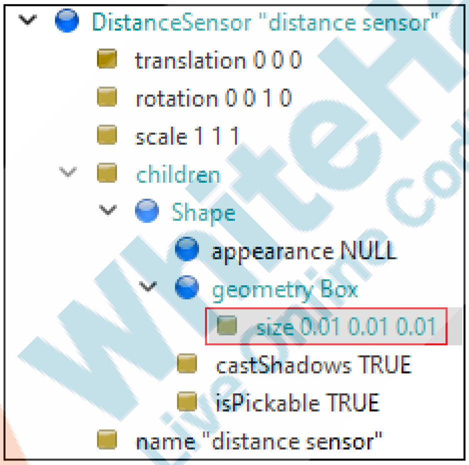


The box which we have just added is pretty big. Let’s change its **dimensions**. For that, let's first **expand** the **geometry** property.



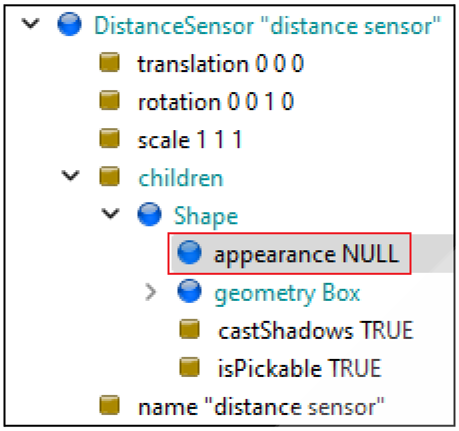
```
DistanceSensor "distance sensor"
    translation 0 0 0
    rotation 0 0 1 0
    scale 1 1 1
    children
        Shape
            appearance NULL
            geometry Box
                size 0.1 0.1 0.1
            castShadows TRUE
```

You will see the **size** property. Change the **size** to **0.01 m** in **x, y** and **z direction**.



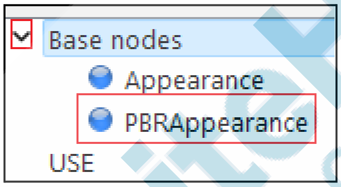
```
DistanceSensor "distance sensor"
    translation 0 0 0
    rotation 0 0 1 0
    scale 1 1 1
    children
        Shape
            appearance NULL
            geometry Box
                size 0.01 0.01 0.01
            castShadows TRUE
            isPickable TRUE
    name "distance sensor"
```

Now once the dimensions are adjusted it's time to select an **appearance** for the **distance sensor**. For that, let's first **double click** on the appearance property.



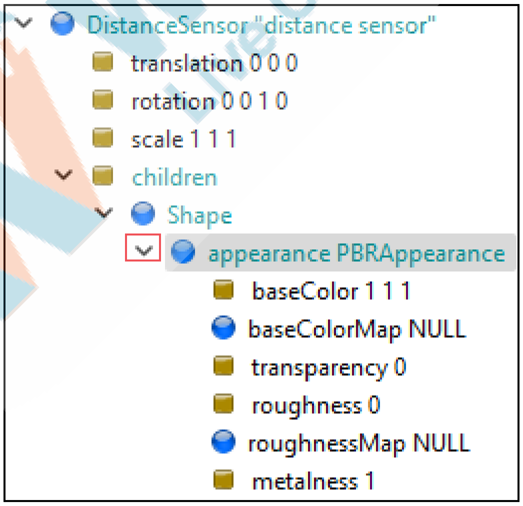
```
DistanceSensor "distance sensor"
  translation 0 0 0
  rotation 0 0 1 0
  scale 1 1 1
  children
    Shape
      appearance NULL
      geometry Box
      castShadows TRUE
      isPickable TRUE
  name "distance sensor"
```

A window will appear. **Expand** the **Base nodes** and select the **PBR Appearance** node.



```
Base nodes
    Appearance
    PBRAppearance
USE
```

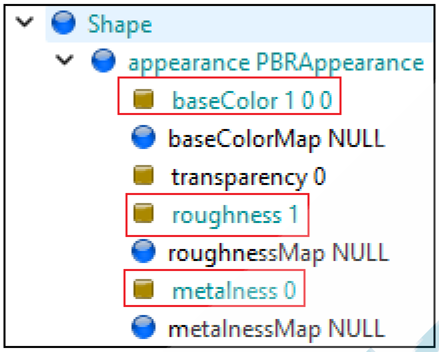
Next, expand the **PBR Appearance** node to change its properties.



```
DistanceSensor "distance sensor"
  translation 0 0 0
  rotation 0 0 1 0
  scale 1 1 1
  children
    Shape
      appearance PBRAppearance
        baseColor 1 1 1
        baseColorMap NULL
        transparency 0
        roughness 0
        roughnessMap NULL
        metalness 1
```

Let's set the **color** of the **distance sensor** as **red**. For that,

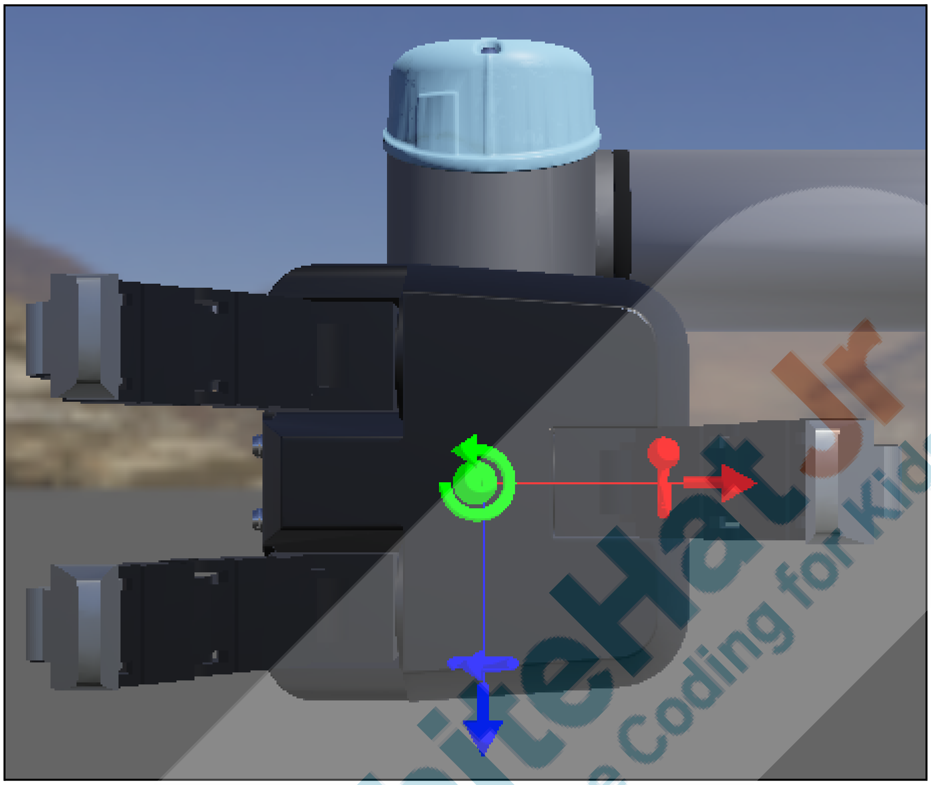
a) Change the **base color** property. Set **1** for **red, 0** for **green** and **0** for **blue**.
b) Set the **roughness** to **1** and **metalness** to **0**.



```
Shape
  appearance PBRAppearance
    baseColor 1 0 0
    baseColorMap NULL
    transparency 0
    roughness 1
    roughnessMap NULL
    metalness 0
    metalnessMap NULL
```

Now there are two more challenges,

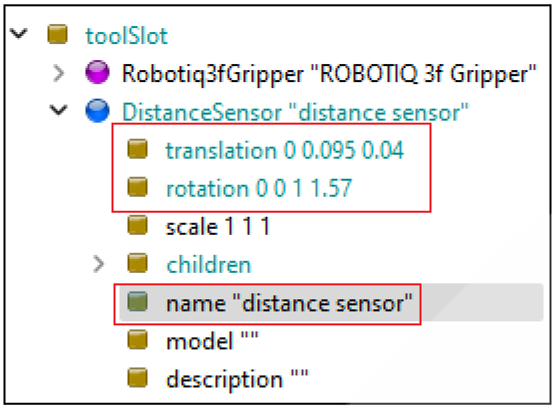
1) The **distance sensor** is **inside** the **gripper hand.**
2) The **axis** of the **distance sensor** is not aligned correctly. The **distance sensor** is **active** or **measures distance** in the **X direction (red axis)**.



To solve the first challenge, let's change the **translation** of the distance sensor node to **0** in **x** direction, **0.095** in **y** direction and **0.04** in the **z** direction. This will make sure that the distance sensor **comes out** of the gripper hand.

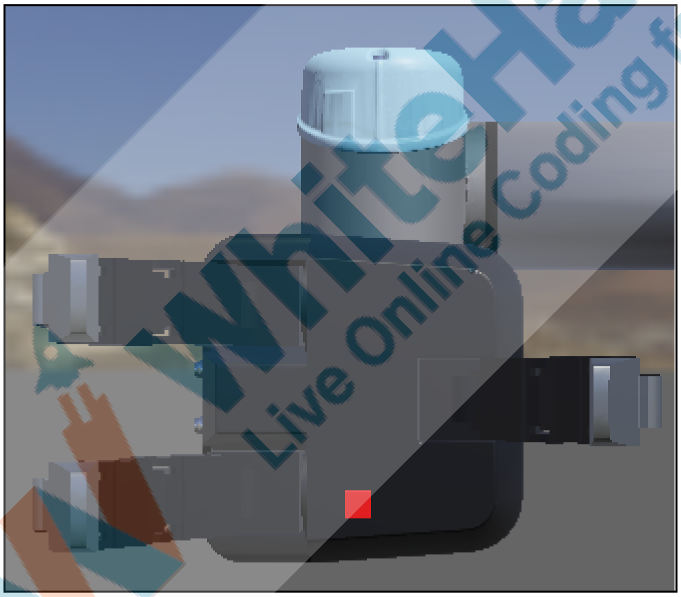
To solve the second challenge, **rotate** the distance sensor by **90 degrees** or **1.57 radians** in **z** direction. For that, we will have to change the **rotation property** of the distance sensor node as, **0** in **x** direction, **0** in **y** direction, **1** in **z** direction and angle as **1.57 radians**.

Finally, **hit enter** and **save** your work till here.

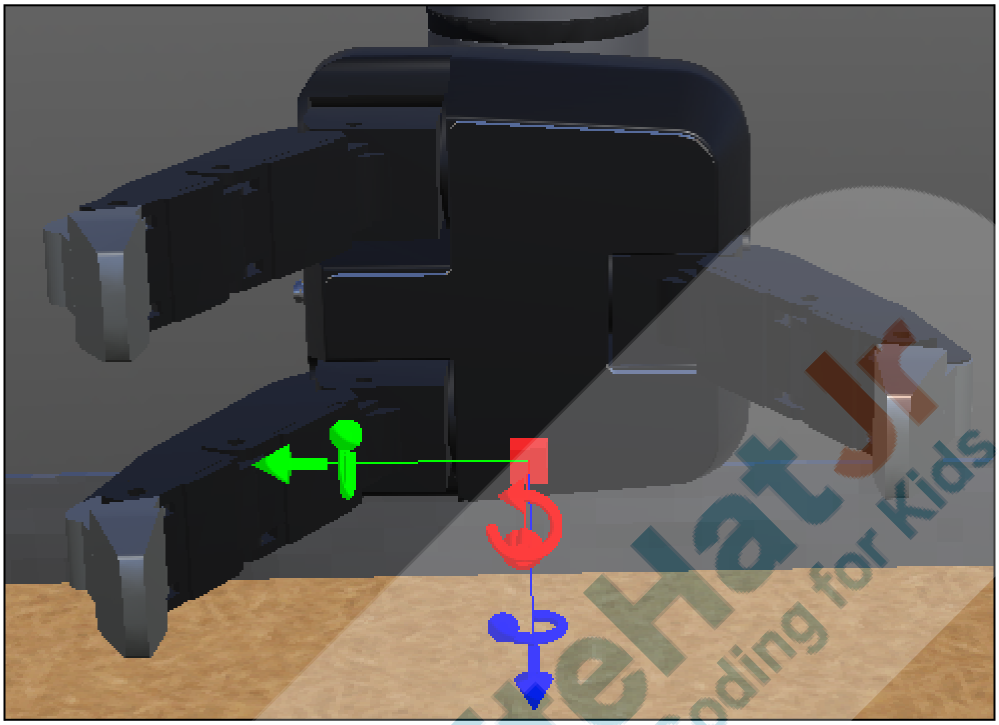


```
toolSlot
  Robotiq3fGripper "ROBOTIQ 3f Gripper"
  DistanceSensor "distance sensor"
    translation 0 0.095 0.04
    rotation 0 0 1 1.57
    scale 1 1 1
    children
    name "distance sensor"
    model ""
    description ""
```

After adjusting the **translation property**, your distance sensor will look as shown in the graphic.



After adjusting the **rotation property**, the **axes** of the distance sensor will look as shown in the graphic.



Teacher Stops Screen Share

So now it’s your turn. Please share your screen with me.	

STUDENT-LED ACTIVITY 15 mins

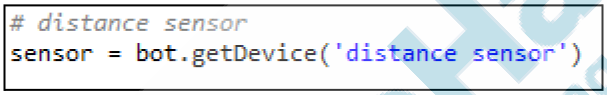
- **Ask the student to press the ESC key to come back to the panel.**
- **Guide the student to start Screen Share.**
- **The teacher gets into Full Screen.**

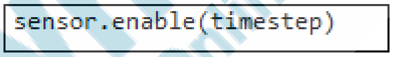
Student Initiates Screen Share

ACTIVITY

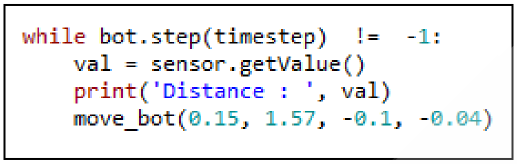
- **Getting sensor values from the sensor.**
- **Picking the solid box from the conveyor and dropping it into a metal storage box.**

Teacher Action	Student Action
Once we are done with the design part. Let's write some code, so that we can capture data from the **distance sensor**, and thus, **detect** the objects which are in front of it. For that, first let's open the student boilerplate link and download all the files from it. Open all the downloaded files in the webots software. Next, in the controller program, let's create an **object** named **sensor** for the distance sensor node using the **.getDevice()** method of the **robot** class.	


`# distance sensor`
`sensor = bot.getDevice('distance sensor')`	

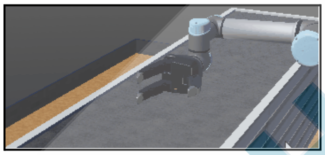
Next, let's enable the sensor using the **.enable()** method. The **enable()** method takes the **controller timestep** as the argument.	
 `sensor.enable(timestep)`	
Next, let's create our **controller loop**, where we can check the **sensor values** as, a) Use the **.step()** method of the Robot class, as the condition of our **while** loop. b) Within the loop, use the **.getValue()** method to get distance values from the sensor. c) **Print** the **distance** so that we can use it to frame our **conditional statements** later. d) Change the position of the **UR5e robotic arm** using the **move_bot()** method, in such a way, so that it goes over the **conveyor belt,** facing towards the **solid objects**. Give the positional arguments to the **move_bot()** method as, ● **Shoulder lift position = 0.15** ● **Shoulder pan position = 1.57**	

- **Elbow position = -0.1**
- **Wrist 1 position = -0.04**
- Other arguments are equal to their default values.

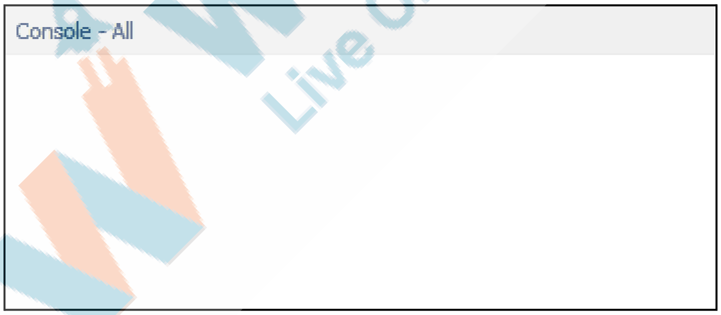


```
while bot.step(timestep)  !=  -1:
    val = sensor.getValue()
    print('Distance : ', val)
    move_bot(0.15, 1.57, -0.1, -0.04)
```

Save your work and run the simulation.



- You will see that if the **red solid box** is not in the range of the **sensor**, we will get a value of **1000** on the **console**.
- Once the box is in the range of the sensor, the distance will start to **decrease** as the **box** moves **towards** the **gripper hand** or the **sensor**.



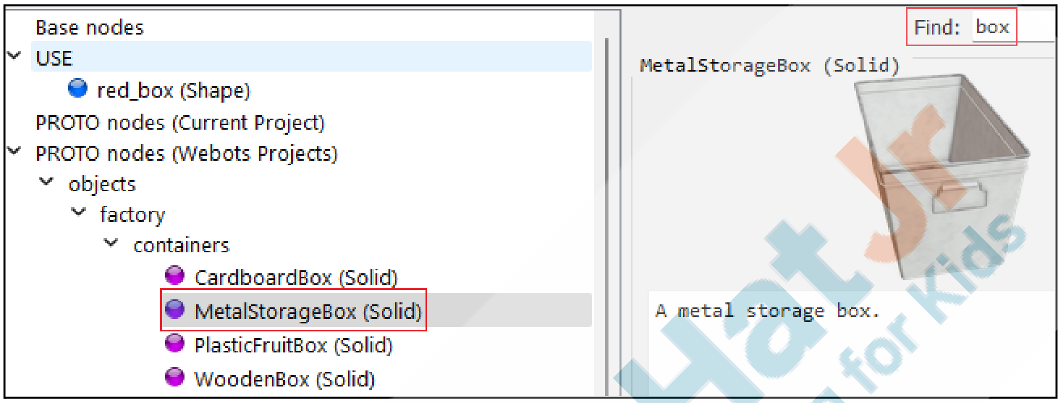
Once we have our distance sensor up and running, let’s bring a **metallic container**, which will hold all the **solid boxes** that will be **picked** and **dropped** by the **UR5e** bot.

For that,

a) Click on the **add object** button.
b) Search the word **‘box’** in the **Find textbox**.

c) You will find a **metal storage box (solid)** listed under **proto nodes** as,
PROTO nodes → objects → factory → containers → Metal storage box (Solid)

d) **Double click** on it, to get it in our environment.



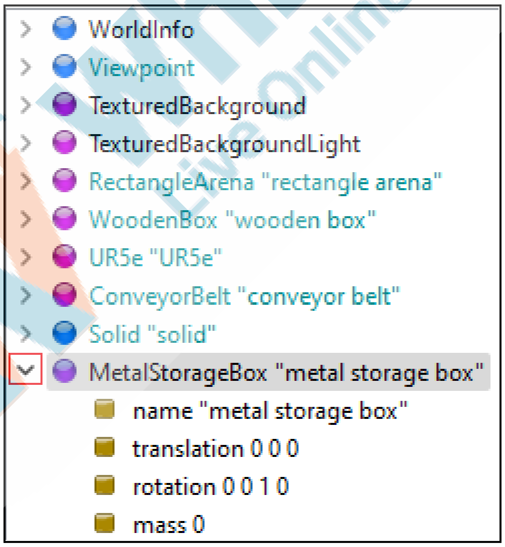
Base nodes
USE
red_box (Shape)
PROTO nodes (Current Project)
PROTO nodes (Webots Projects)
objects
factory
containers
CardboardBox (Solid)
MetalStorageBox (Solid)
PlasticFruitBox (Solid)
WoodenBox (Solid)

Find: box

MetalStorageBox (Solid)

A metal storage box.

You will see a **metal storage box node** added in the **scene tree**. But we will have to reposition it. For that, let's first **expand** it.

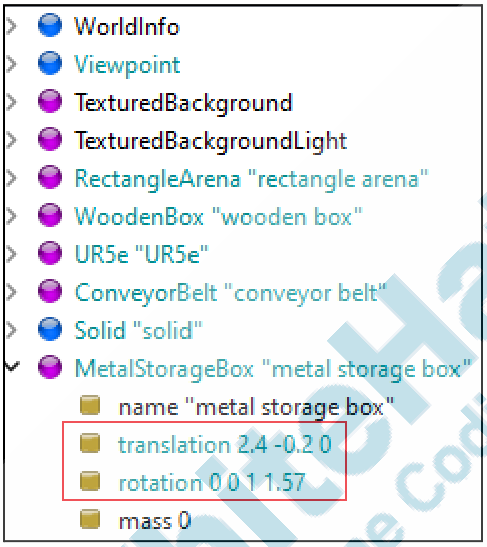


WorldInfo
Viewpoint
TexturedBackground
TexturedBackgroundLight
RectangleArena "rectangle arena"
WoodenBox "wooden box"
UR5e "UR5e"
ConveyorBelt "conveyor belt"
Solid "solid"
MetalStorageBox "metal storage box"
name "metal storage box"
translation 0 0 0
rotation 0 0 1 0
mass 0

Once you expand the node, you will see the **translation** and the **rotation properties** of the **metal storage box**.

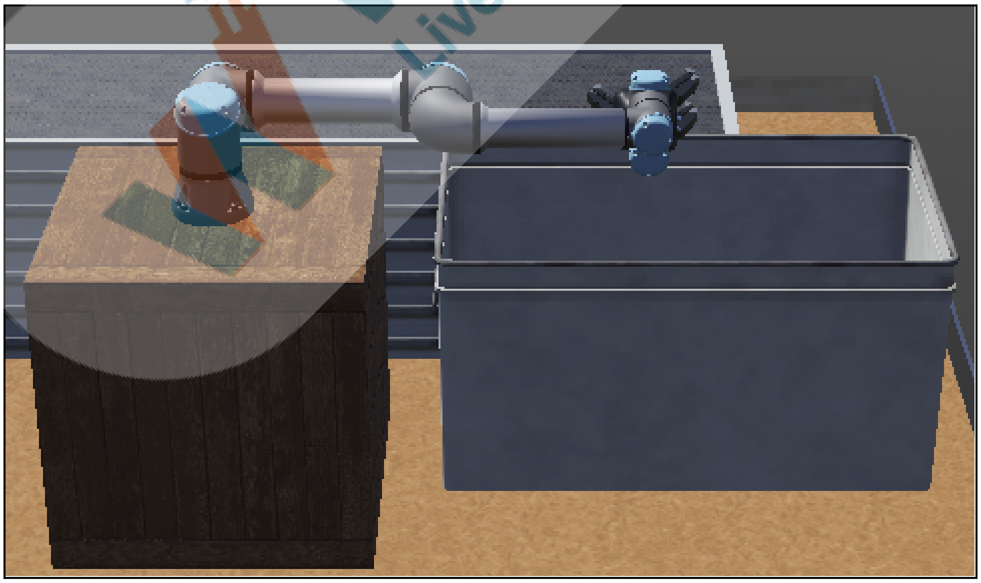
Change its **translation** to **2.4** in **x** direction, **-0.2** in **y** direction and **0** in **z** direction.

After changing the translation, change its **rotation** to **0** in **x** direction, **0** in **y** direction, **1** in **z** direction and specify the **rotation angle** as **1.57 radians** (or **90 degrees**).



```
> WorldInfo
> Viewpoint
> TexturedBackground
> TexturedBackgroundLight
> RectangleArena "rectangle arena"
> WoodenBox "wooden box"
> UR5e "UR5e"
> ConveyorBelt "conveyor belt"
> Solid "solid"
v MetalStorageBox "metal storage box"
    name "metal storage box"
    translation 2.4 -0.2 0
    rotation 0 0 1 1.57
    mass 0
```

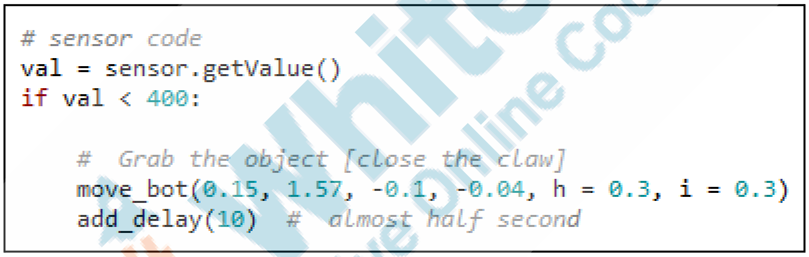
Save your work till here. You will see the box repositioned as shown in the graphic.



Once we are done with the design part, it’s time to write some code, so that the **UR5e robotic arm** can **pick** things up from the **conveyor** and **drop** them in the **metal storage box**.

For that,

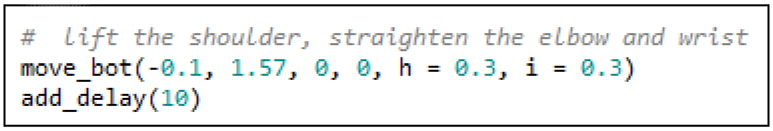
a) Let’s get the value from the sensor using the **.getValue()** method.
b) Check if the **distance** between the **solid box** and **gripper hand** (or the sensor) is **less** than **400** or not.
c) If yes, first **grab** the object by changing the positions of the **lower knuckle** and the **middle knuckle** to **0.3 radians**.
d) Wait for **10 timesteps** or **10 * 64 = 640 ms**, before moving the UR5e arm to the other place, using the **add_delay()** method.



```
# sensor code
val = sensor.getValue()
if val < 400:

    #  Grab the object [close the claw]
    move_bot(0.15, 1.57, -0.1, -0.04, h = 0.3, i = 0.3)
    add_delay(10)  #  almost half second
```

Once you have grabbed the object,

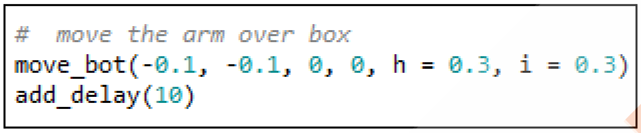
a) **Lift** the **shoulder** joint to **-0.1 radians**.
b) **Move** the **elbow** and **wrist joints** to **0 radians**.
c) Keep the gripper hand closed. Don’t change the position for the lower and middle knuckles.
d) Wait for **10** more **timesteps**, before moving the arm to another position.



```
#  lift the shoulder, straighten the elbow and wrist
move_bot(-0.1, 1.57, 0, 0, h = 0.3, i = 0.3)
add_delay(10)
```

Once you have lifted the solid box,

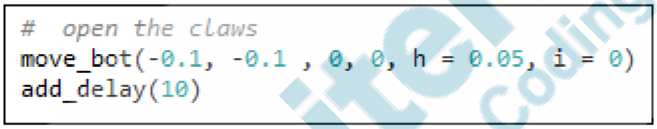
a) **Move** or **pan** the **shoulder** joint to **-0.1 radians**, so that the arm is over the **metal storage box**.
b) Don't change any other positions.
c) Wait for **10** more **timesteps**, before moving the arm to another position.



```
#  move the arm over box
move_bot(-0.1, -0.1, 0, 0, h = 0.3, i = 0.3)
add_delay(10)
```

Next, **open** the **claws** by changing the position of **lower knuckle** to **0.05 radians** and **middle knuckle** to **0 radians**.

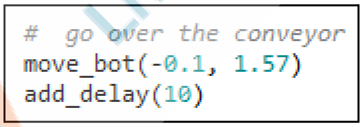
After that, **wait** for **10** more **timesteps**.



```
#  open the claws
move_bot(-0.1, -0.1 , 0, 0, h = 0.05, i = 0)
add_delay(10)
```

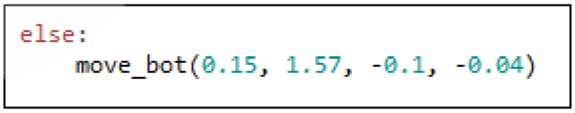
Finally, go over the conveyor by changing the **shoulder pan position** to **1.57 radians.**

Wait for **10** more **timesteps**.



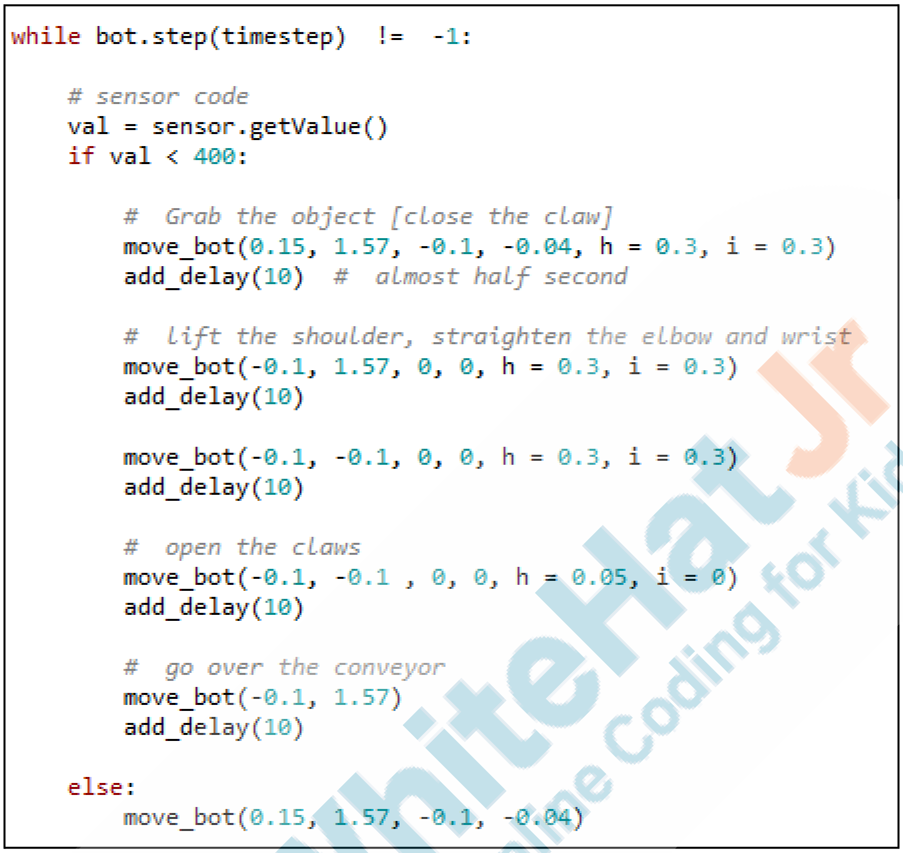
```
#  go over the conveyor
move_bot(-0.1, 1.57)
add_delay(10)
```

If there is no object in front of the UR5e robotic arm, stay over the conveyor waiting for the solid box.



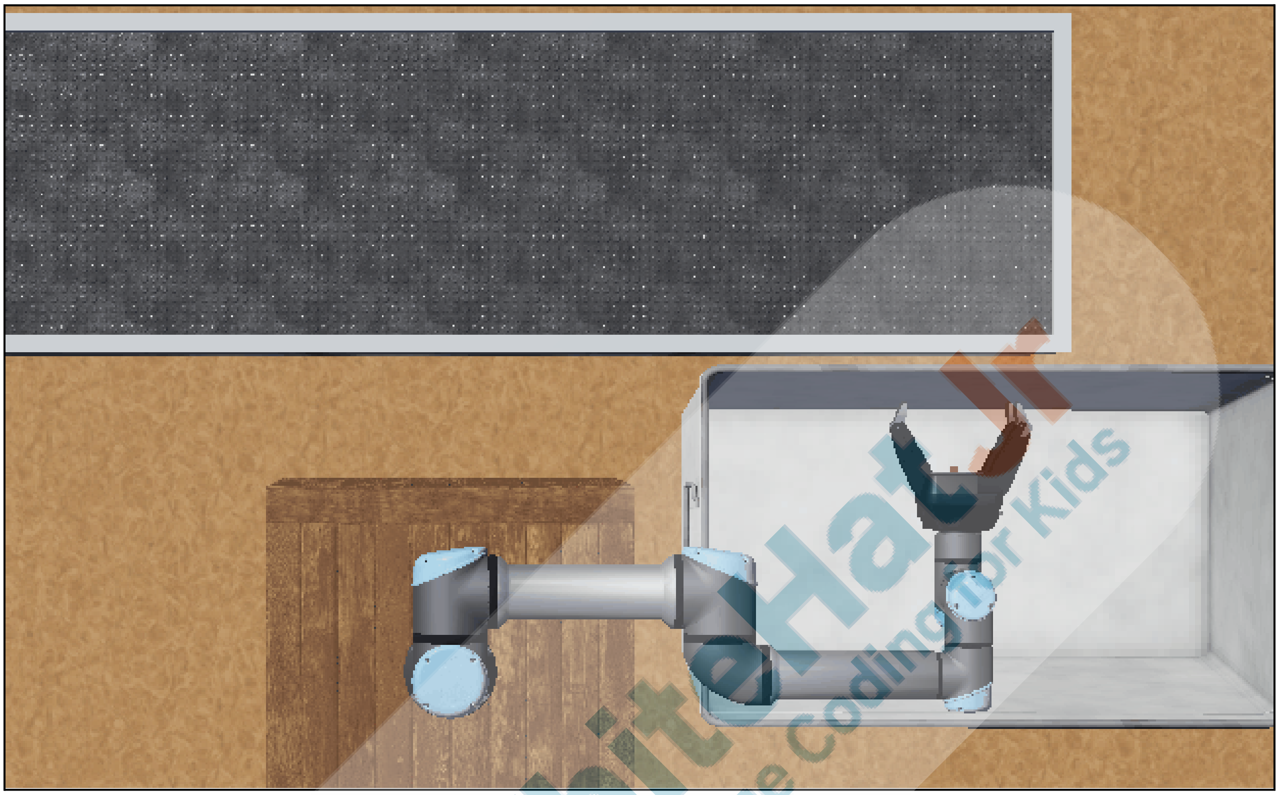
```
else:
    move_bot(0.15, 1.57, -0.1, -0.04)
```

The complete code would look as shown in the graphic.



```
while bot.step(timestep)  !=  -1:

    # sensor code
    val = sensor.getValue()
    if val < 400:

        #  Grab the object [close the claw]
        move_bot(0.15, 1.57, -0.1, -0.04, h = 0.3, i = 0.3)
        add_delay(10)  #  almost half second

        #  lift the shoulder, straighten the elbow and wrist
        move_bot(-0.1, 1.57, 0, 0, h = 0.3, i = 0.3)
        add_delay(10)

        move_bot(-0.1, -0.1, 0, 0, h = 0.3, i = 0.3)
        add_delay(10)

        #  open the claws
        move_bot(-0.1, -0.1 , 0, 0, h = 0.05, i = 0)
        add_delay(10)

        #  go over the conveyor
        move_bot(-0.1, 1.57)
        add_delay(10)

    else:
        move_bot(0.15, 1.57, -0.1, -0.04)
```

Save your work till here and run the simulation.	



Click here to view the reference video.

Teacher Guides Student to Stop Screen Share

WRAP-UP SESSION - 5 mins

Activity details

Following are the WRAP-UP session deliverables:

- Appreciate the student.
- Revise the current class activities.
- Discuss the quizzes.

WRAP-UP QUIZ
Click on In-Class Quiz

Activity Details

Following are the session deliverables:

- Explain the facts and trivia
- Next class challenge
- Project for the day
- Additional Activity (Optional)

FEEDBACK

- **Appreciate and compliment the student for trying to learn a difficult concept.**
- **Get to know how they are feeling after the session.**
- **Review and check their understanding.**

Teacher Action	
You get “hats-off” for your excellent work! In the next class, we will add a camera to our gripper hand so that we can create an object sorting system.	

PROJECT OVERVIEW DISCUSSION

Refer the document below in Activity Links Sections

Teacher Clicks ✖ End Class

ACTIVITY LINKS

Activity Name	Description	Links
Teacher Reference 1	Boilerplate code	https://github.com/procodingclass/PRO-C290-Teacher-Boilerplate.git
Teacher Activity 1	Reference Code	https://github.com/procodingclass/PRO-C290-Reference-Code.git
Teacher Reference 1	Project	https://s3-whjr-curriculum-uploads.

		whjr.online/55bc2ed1-9354-4bb8-8f5b-7dd8feadea31.pdf
Teacher Reference 2	Project Solution	https://github.com/procodingclass/PRO-C290-Project-Solution
Teacher Reference 4	In-Class Quiz	https://s3-whjr-curriculum-uploads.whjr.online/50cfce55-7044-4ee6-b8d4-19ea2cdd33d8.pdf
Student Activity 1	Boilerplate Code	https://github.com/procodingclass/PRO-C290-Student-Boilerplate.git
Teacher Reference 5	Final output gif	https://s3-whjr-curriculum-uploads.whjr.online/f2145c44-007a-4ace-94f0-9ae2c741f68d.gif