Topic	ROOM CLEANING ROBOT- I	
Class Description	**Students will start working with an automated room cleaning robot. They will also learn about IMU sensors and understand its applications.**	
Class	**PRO C296**	
Class time	**45 mins**	
Goal	• Learning about automated cleaning robot • Understanding IMU 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	

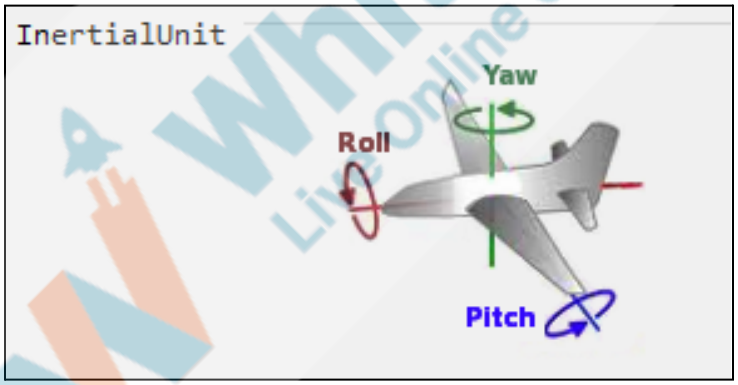
Class structure	**Warm-Up** **Teacher-Led Activity 1** **Student-Led Activity 1** **Wrap-Up**	**10 mins** **10 mins** **20 mins** **05 mins**
Credit & Permissions:	**This project uses Webots, an open-source mobile robot simulation software developed by Cyberbotics Ltd.** **License**	
WARM-UP SESSION - 10 mins		
Teacher Action		**Student Action**

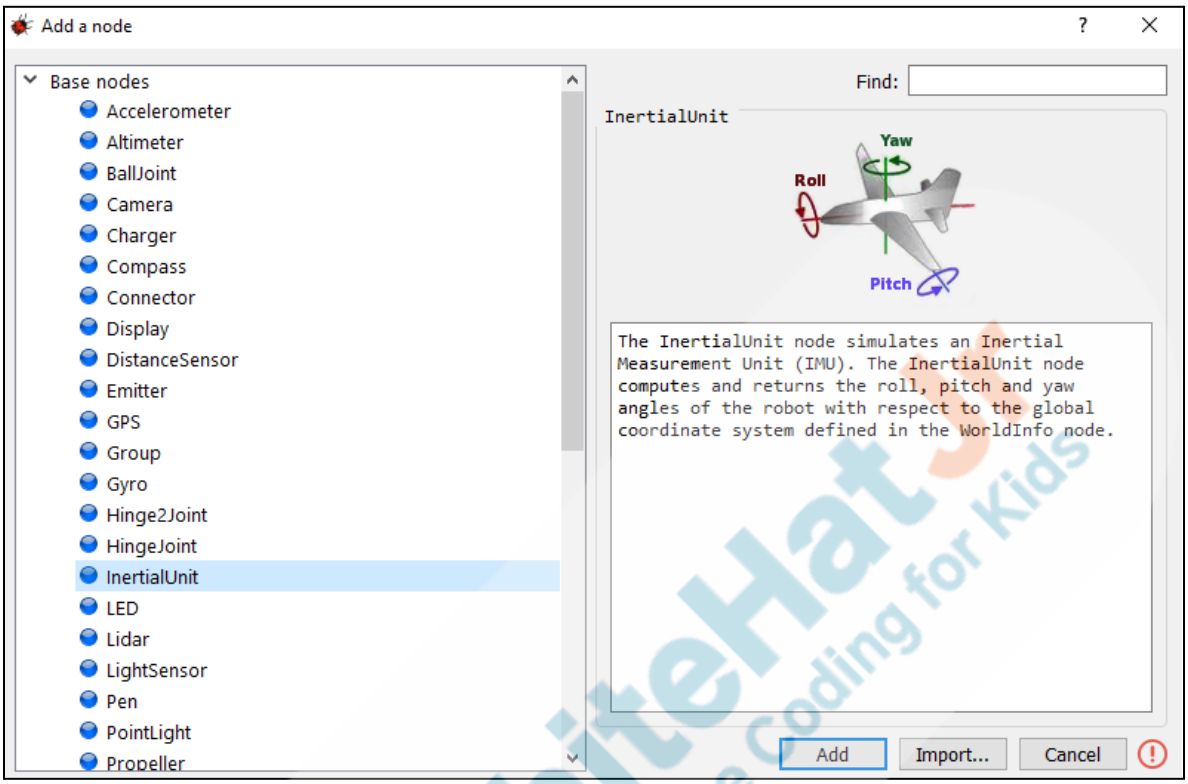
Hey <student's name>. How are you? It's great to see you! Are you excited to learn something new today? **Following are the WARM-UP session deliverables:** • Greet the student. • Revision of previous class activities. • Quizzes.	**ESR**: Hi, thanks! Yes I am excited about it! Click on the slide show tab and present the slides
WARM-UP QUIZ Click on In-Class Quiz	
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.	
TEACHER-LED ACTIVITY - 10 mins	
Teacher Initiates Screen Share	
• **Adding a cleaning robot to the simulator.** • **Understanding the IMU sensor.**	
Teacher Action	**Student Action**
So what did we learn in our last class? *The teacher will clarify if there are any doubts!* So let's get started with today's class.	**ESR:** Varied!
Note: Make sure you save the project after every major step. Otherwise, you might	

lose your work.

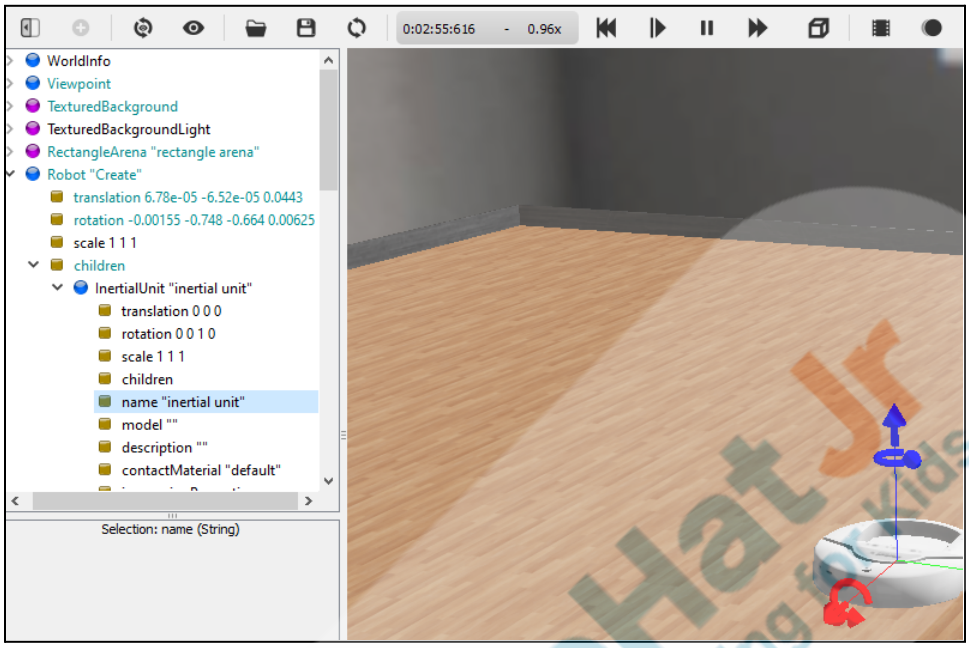
Do you help around at home with the daily chores?	**ESR:** yes.
Which is the one chore at home that you don't like to do?	**ESR:** Varied!
Our robot should make our life easier. It's especially good if it can help us with our daily chores.	
Today, we are going to work with a robot which can clean the floor. It is a robot vacuum cleaner which will help keep our homes clean without much human effort. Let's start with our project. *The teacher downloads the boilerplate code from Teacher Activity 1.* Let's get started with a new project. 1. Open **Webots** Robot Simulator. 2. Open the downloaded project.	

In the boilerplate code, the floor and the environment is already present for us. The room cleaning robot is also added. Let's observe this robot. [The room cleaning robot is a pre-built robot. You can also find this room cleaning robot by clicking on the **Add New** button → Go to **PROTO nodes (Webots Projects)** → **robots** → **irobot**→ **create**→ **Create(Robot)** In the boilerplate code, the controller of the robot was reset to none. Like other prebuilt robots, this also comes with a pre-written controller. But in this case, the controller is written in the programming language C and it randomly chooses directions while moving around. We will write our controller in python and will take calculated turns using the IMU sensor.]	
Let's start with our project. How do you think the robot should move around in the room? Great! We can go ahead with the back and forth motion. 	**ESR:** In a way that it covers the whole room.

But to achieve this motion, we need to take precise turns. How do you think we can do that?	**ESR:** varied.
We can achieve these precise turns by using an **InertialUnit**. The InertialUnit node simulates an Inertial Measurement Unit (IMU). The **InertialUnit** node computes and returns the **roll**, **pitch** and **yaw** angles of the robot with respect to the global coordinate system defined in the WorldInfo node. Here, **yaw** - nose left or right rotation about the vertical axis. **pitch** - nose up or down about the side-to-side axis. **roll** - rotation around the front-to-back axis. By observing these values, we can steer the robot in the desired direction.	
	
1. Add an **InertialUnit** to the children of the robot.	



2. We can see the name of the new inertial unit is set as “inertial unit”. We will use this name to refer to the device in the controller.



3. Create a new controller named "**robo_controller**"and assign it as the controller property for this room cleaning robot.

4. In the controller, first import necessary modules, create an instance of robot class and initiate the timestep variable.

```
from controller import Robot

robot = Robot()

timestep = int(robot.getBasicTimeStep())
```

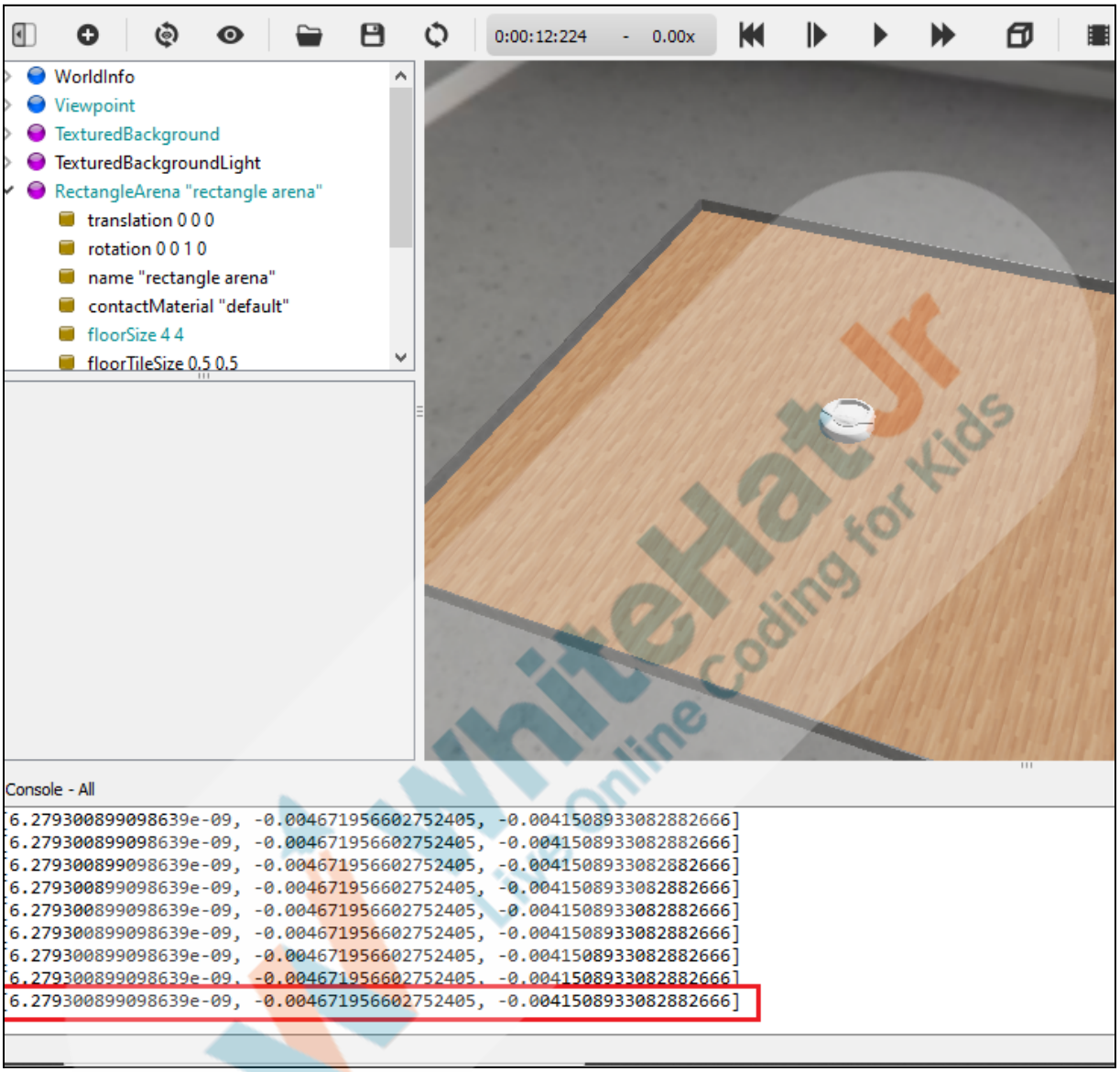
5. Use **getDevice()** method to create a unique identifier for the InertialUnit. Once that is done, enable the device.

```
imu= robot.getDevice("inertial unit")
imu.enable(timestep)
```

6. Now, we will write the main while loop. We need to read the **roll**, **pitch** and **yaw** value from the inertial unit. For this, we have a method named **getRollPitchYaw().**

```
1  from controller import Robot
2
3  robot = Robot()
4
5  timestep = int(robot.getBasicTimeStep())
6
7  imu= robot.getDevice("inertial unit")
8  imu.enable(timestep)
9
10 while robot.step(timestep) != -1:
11
12     angle=imu.getRollPitchYaw()
13     print(angle)
14
```

Now, if we run the code, we can see that the **getRollPitchYaw()** returns three values in a list - **roll**,**pitch**, **yaw** in that order.



In this case, we want to take turns towards left and right. So, which value do you think we will need? Yaw is the third value in the returned list. So, the index of yaw should be 2.	**ESR:** We will use the yaw value.

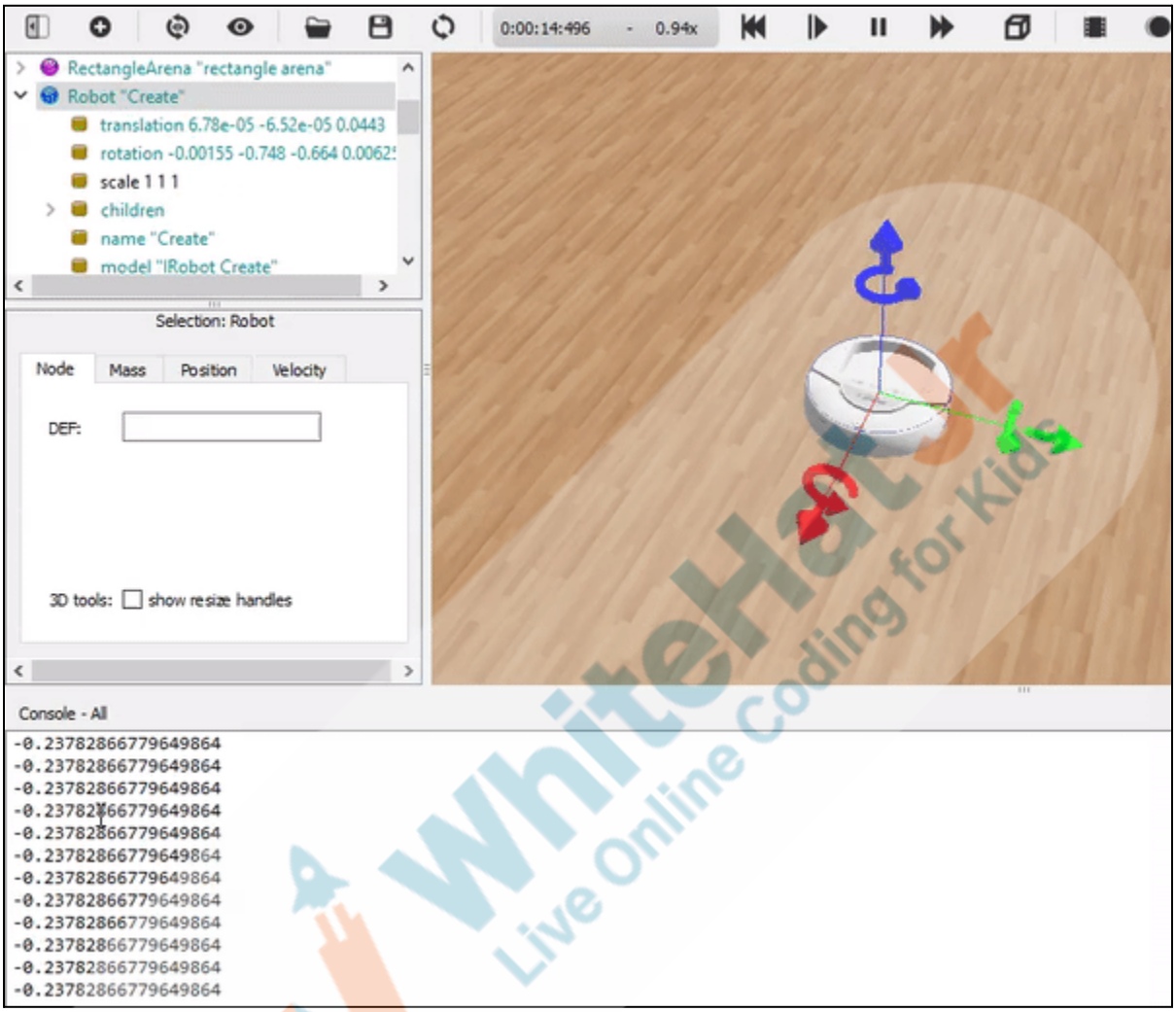
```
yaw_current= angle[2]
```

The angle value returned by the **getRollPitchYaw()** method is measured in radians. We can use the **math.degrees()** method to convert the angle from radians to degrees.

```
yaw_current= math.degrees(angle[2])
```

Make sure to import the **math** module at the beginning.

```
2 import math
```



We can see that the angle value varies from -180 degrees to +180 degrees.

As we can see, it's a floating point number, we need to round it off for our convenience.

```
yaw_current= round(math.degrees(angle[2]))
```

To make our task easier let's add 180 to yaw value. This will make it range between 0 to 360 degrees.

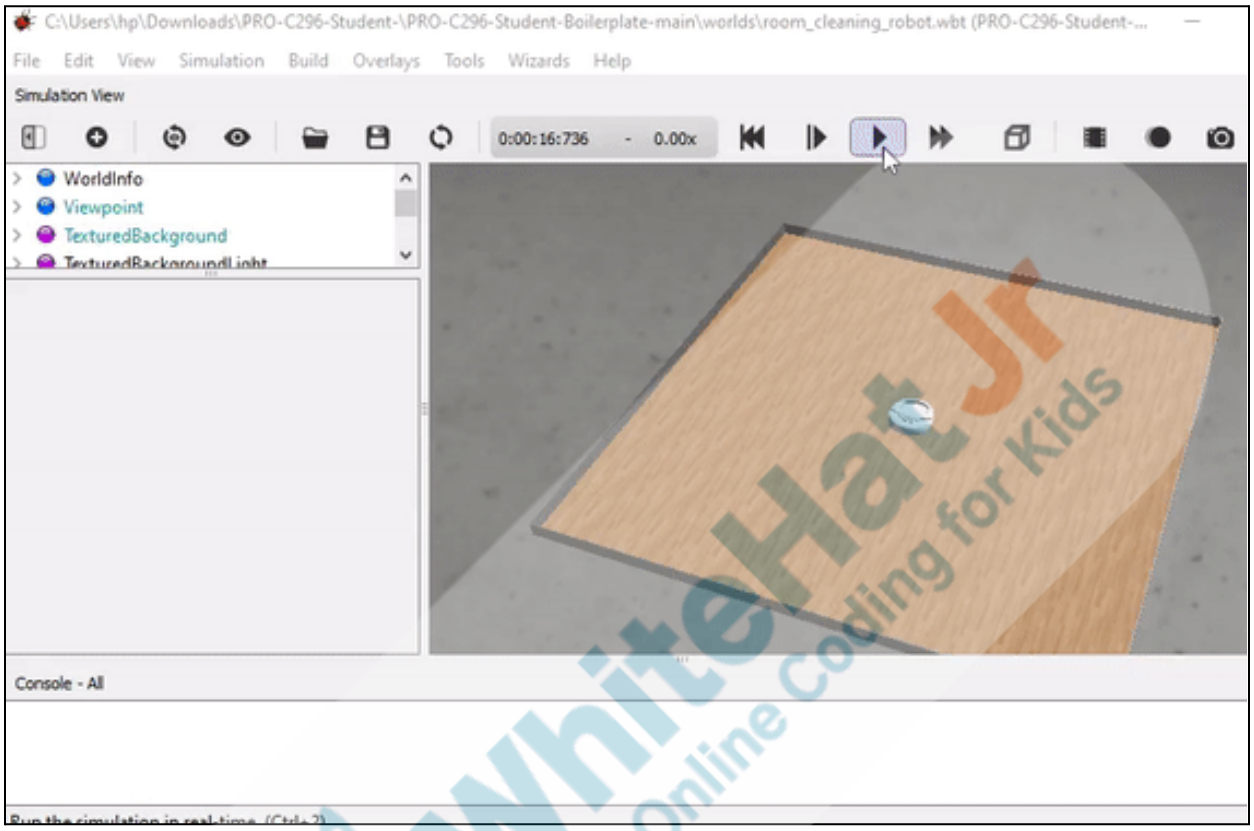
```
yaw_current= round(math.degrees(angle[2]))+180
```

Reference Code:

robo_controller.py

```
from controller import Robot
import math

robot = Robot()

timestep = int(robot.getBasicTimeStep())

imu= robot.getDevice("inertial unit")
imu.enable(timestep)


while robot.step(timestep) != -1:

    angle=imu.getRollPitchYaw()

    yaw_current= round(math.degrees(angle[2]))+180

    print(yaw_current)
```

Reference Output:



Now, you will code using this data from the Inertial Unit , you will try to write code which will help the robot take precise turns.	
Are you ready?	**ESR:** Yes.
Teacher Stops Screen Share	
So now it's your turn. Please share your screen with me.	
We have one more class challenge for you. Can you solve it?	

Let's try. I will guide you through it.	
STUDENT-LED ACTIVITY - 20 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 • **Write a method to turn the vehicle to a certain direction.** • **Add distance sensors and write code for wall detection.**	
Teacher Action	**Student Action**
Student downloads the boilerplate code from Student Activity 1.	
The teacher helps the student to write the code. The student takes the lead and tries to figure out how to solve this problem.	
To move this robot, what do you think we need to do? Great! Let's do that first.	**ESR:** Firstly, we need to locate the motors of the wheels and write code to move them.
1. Use the **getDevice()** method to create reference for the two motors - "**left wheel motor**" and "**right wheel motor**".	

robo_controller.py

```
from controller import Robot
import math

robot = Robot()

timestep = int(robot.getBasicTimeStep())

left_motor = robot.getDevice("left wheel motor");
left_motor.setPosition(float('inf'))
left_motor.setVelocity(0.0)

right_motor = robot.getDevice("right wheel motor");
right_motor.setPosition(float('inf'))
right_motor.setVelocity(0.0)

imu= robot.getDevice("inertial unit")
imu.enable(timestep)

while robot.step(timestep) != -1:

    angle=imu.getRollPitchYaw()

    yaw_current= math.degrees(angle[2])+180
    print(yaw_current)
```

2. Now, define a method named **move()**. This method will have two parameters - **left_speed**, **right_speed**. We will use this method whenever we need to move the robot.

```
def move(left_speed,right_speed):
    left_motor.setVelocity(left_speed)
    right_motor.setVelocity(right_speed)
```

3. Now, the challenging part is to turn our robot towards a certain angle. We know our current orientation. Using the **yaw_current** value, we will turn our robot towards the desired direction. What do you think should be the logic for this portion of code? Okay. We know how to turn the robot, right? Exactly. Here, we want the robot to stay at its current position and turn around its center. It is called a point turn. We need to pass the same speed for both the motors, but one should be positive and another negative. E.g. **move(-1, 1)** We will keep the values small so that the turns are slower and precise.	**ESR:** Varied. **ESR:** Yes. We can call the **move()** method and pass different values for the left and right motor's speed so that it turns.

```
if(90 != yaw_current):
    move(-1,1)
else:
    move(0,0)
```

Let's make this function so that we can use it whenever we want.

```
def turn_towards_angle(target_angle):
    if(target_angle != yaw_current):
        move(-1,1)
    else:
        move(0,0)
```

We will also need a method which will move the robot forward until it detects an obstacle.

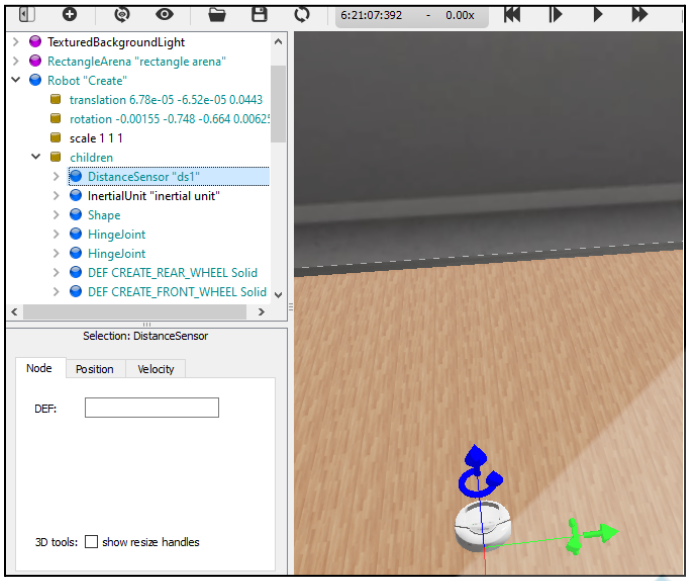
Can you tell me what we should use to detect obstacles?

ESR: We can use distance sensors.

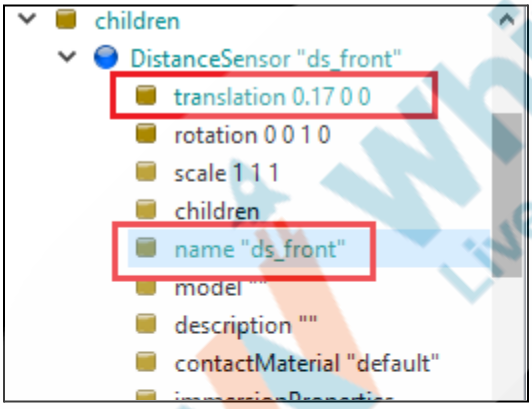
Correct! We can use distance sensors.

For this new method, we need one distance sensor mounted at the front of the robot.

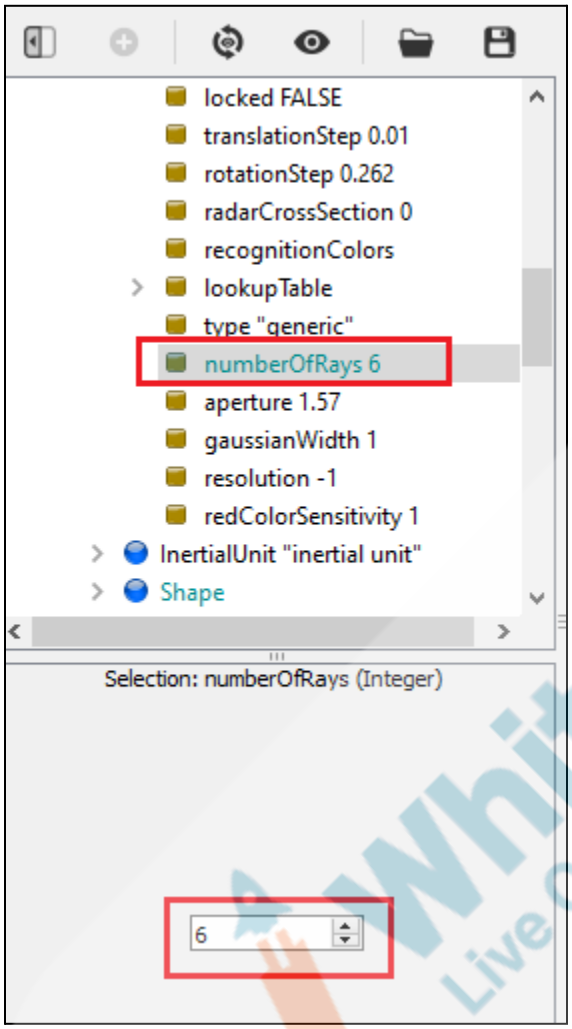
Later in this project, we will also need to detect if there is any obstacle towards the left or right of the robot. So, we will add three distance sensors on our robot.



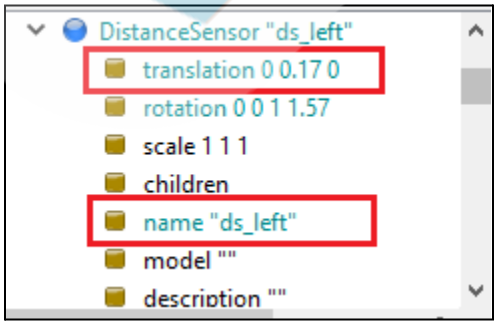
1. Change the name to "**ds_front**" and change the **translation** property to **x - 0.17, y - 0, z - 0**.

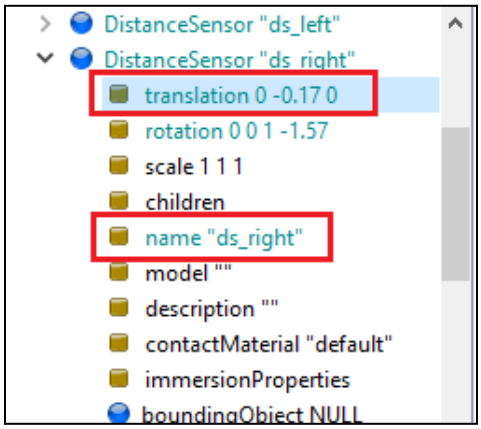


2. You can also increase the number of rays of the distance sensor for better detection of obstacles.



Similarly, add two more distance sensors to the left and right of the robot.





Now, let's write the method which will move the robot forward until it detects an obstacle.

1. Use **getDevice()** method to create unique identifiers for the distance sensors. Once this is done, enable the devices.

```
ds_front = robot.getDevice("ds_front")
ds_front.enable(timestep)
ds_left = robot.getDevice("ds_left")
ds_left.enable(timestep)
ds_right= robot.getDevice("ds_right")
ds_right.enable(timestep)
```

2. Use the **getValue()** method to read value from the distance sensor **ds_front**.

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

    angle=imu.getRollPitchYaw()

    ds_front_value= ds_front.getValue()

    yaw_current= round(math.degrees(angle[2]))+180

    print(yaw_current)

    move_forward_till_wall_detection()
```

3. Now, write a function which will make the robot move forward until it detects an obstacle with the "**ds_front**" distance sensor.

```
35
36 def move_forward_till_wall_detection():
37     if(ds_front_value>300):
38         move(15,15)
39     else:
40         move(0,0)
41
```

Reference Code:

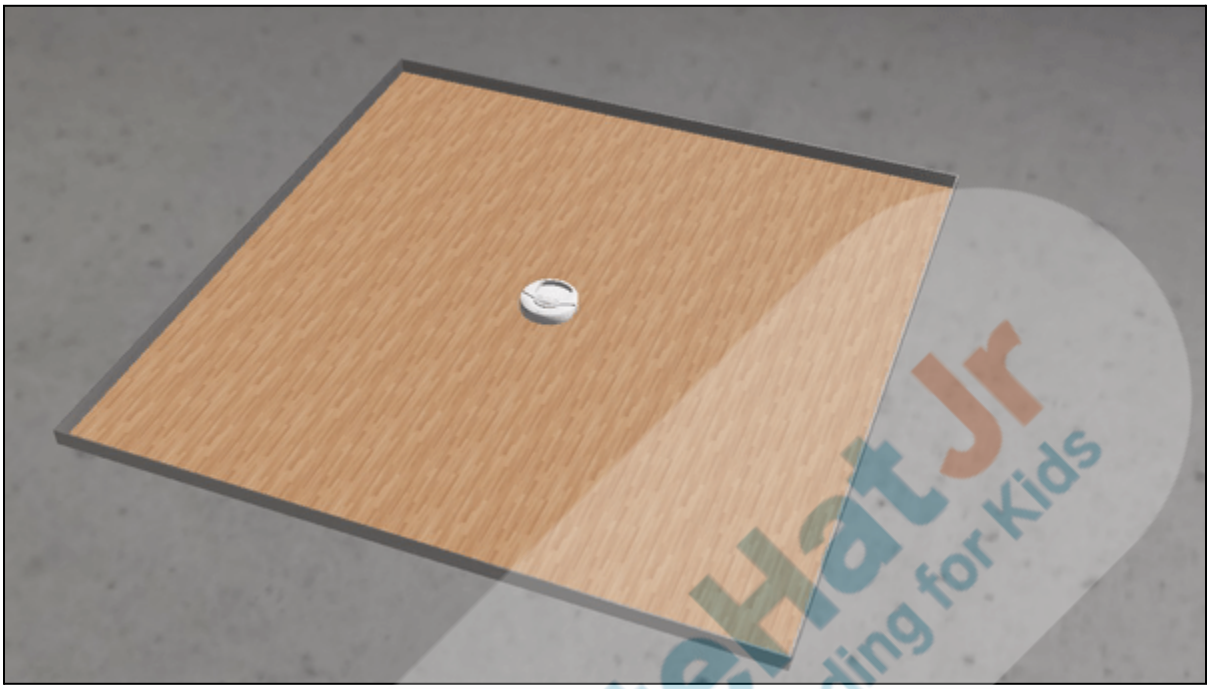
robo_controller.py

```python
from controller import Robot
import math

robot = Robot()

timestep = int(robot.getBasicTimeStep())

left_motor = robot.getDevice("left wheel motor");
left_motor.setPosition(float('inf'))
left_motor.setVelocity(0.0)

right_motor = robot.getDevice("right wheel motor");
right_motor.setPosition(float('inf'))
right_motor.setVelocity(0.0)

imu= robot.getDevice("inertial unit")
imu.enable(timestep)

ds_front = robot.getDevice("ds_front")
ds_front.enable(timestep)
ds_left = robot.getDevice("ds_left")
ds_left.enable(timestep)
ds_right= robot.getDevice("ds_right")
ds_right.enable(timestep)

def move(left_speed,right_speed):
    left_motor.setVelocity(left_speed)
    right_motor.setVelocity(right_speed)

def turn_towards_angle(target_angle):
    if(target_angle != yaw_current):
        move(-1,1)
    else:
        move(0,0)
```

robo_controller.py

```
ds_front.enable(timestep)
ds_left = robot.getDevice("ds_left")
ds_left.enable(timestep)
ds_right= robot.getDevice("ds_right")
ds_right.enable(timestep)

def move(left_speed,right_speed):
    left_motor.setVelocity(left_speed)
    right_motor.setVelocity(right_speed)

def turn_towards_angle(target_angle):
    if(target_angle != yaw_current):
        move(-1,1)
    else:
        move(0,0)

def move_forward_till_wall_detection():
    if(ds_front_value>300):
        move(15,15)
    else:
        move(0,0)

while robot.step(timestep) != -1:

    angle=imu.getRollPitchYaw()

    ds_front_value= ds_front.getValue()

    yaw_current= round(math.degrees(angle[2]))+180

    print(yaw_current)

    move_forward_till_wall_detection()
```

Reference Output:



Click here to view the output video.

Great work!

Teacher Guides Student to Stop Screen Share

WRAP-UP SESSION - 05 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	Student Action
You get “hats-off” for your excellent work! In the next class, we will complete the code for our room cleaning robot.	*Make sure you have given at least 2 hats-off during the class for:* 

PROJECT OVERVIEW DISCUSSION

Refer the document below in Activity Links Sections

Teacher Clicks ✖ End Class

ACTIVITY LINKS		
Activity Name	**Description**	**Links**

Teacher Activity 1	Teacher Boilerplate Code	https://github.com/procodingclass/PRO-C296-Teacher-Boilerplate
Teacher Reference 1	Reference Code	https://github.com/procodingclass/PRO-C296-Reference-Code
Teacher Reference 2	Project	
Teacher Reference 3	Project Solution	
Teacher Reference 4	In-Class Quiz	https://s3-whjr-curriculum-uploads.whjr.online/d4a2d7f1-99ef-43d3-8868-77193d66202d.pdf
Teacher Reference 5	Final Output Reference	https://s3-whjr-curriculum-uploads.whjr.online/e9e32a65-9be5-4a91-8d20-ac25a60e5ca9.mp4
Student Activity 1	Boilerplate Code	https://github.com/procodingclass/PRO-C296-Student-Boilerplate