MIDOCEAN UNIVERSITY

Professional Workshop (1) on Robotics Design, Programming, and AI Integration.

Presented by Dr. Khaled Eskaf

MIDOCEAN UNIVERSITY

- Ph.D in Computer Science from Salford University (UK), Assistant Professor in Intelligent System.

  More than 20 years in Teaching computer science courses, IET STEM Ambassador UK.

- Training, Coaching Students in Robotics,
- Programming Languages (Java, python,Kotlin)
- Game Programming, Virtual Reality, Augmented Reality and Embedded System.
- Judging for many local and national Robotics competitions.

FIRST CANADA

makeblock education

makeblock MAKEX ROBOTICS COMPETITION

MATE ROV COMPETITION

IET STEM AMBASSADORS

Day 1

outline

Day 2

MIDOCEAN UNIVERSITY

Day 1: Introduction to Robotics and Basic Programming

Session 1: Welcome and Introduction
Session 2: Basic Components of Robots
Session 3: Motors and Actuators
Session 4: Sensor Integration

Session 5: Introduction to Webots
- Overview of Webots Simulation Environment
- Setting Up Your First Webots Project
- Creating and Running Simple Robot Controllers

Hands-on Examples:
1. Obstacle Detection and LED Flashing with e-puck Robot
2. Capturing and Analyzing Images with e-puck Robot's Camera
3. Red Obstacle Detection and Navigation with e-puck Robot
4. Color Score-Based Speed Control for e-puck Robot

Day 2: Advanced Applications and Machine Learning

Session 1: Recap and Q&A
Session 2: Advanced Navigation Techniques
- Using Multiple Sensors for Enhanced Obstacle Avoidance
- Combining Sensor Data for Complex Behaviors

Session 3: Introduction to Machine Learning in Robotics
- Applying Linear Regression to Robot Control

Session 4: Practical Machine Learning Applications
- Implementing Control Algorithms for Specific Tasks
- Real-time Data Processing and Decision-Making

Hands-on Examples:
1. Distance-Based Speed Control with Linear Regression for e-puck Robot
2. Avoiding Obstacles Using Multiple Infrared Sensors
3. Robot that Flashes its LED and Avoids Obstacles
4. E-puck Camera with Color Object Detection

MIDOCEAN UNIVERSITY

Who is the Father of robotics ?

1- George Devol

2- Ismail al-Jazari

3- Isaac Newton

Day
01

Who is the Father of robotics ?

MIDOCEAN UNIVERSITY

1- George Devol

Devol's work in the 1950s and 1960s laid the groundwork for the robotics industry as we know it today

2- Ismail al-Jazari

"Father of Robotics." Al-Jazari was a 12th-century Islamic scholar and inventor who created numerous automatons and mechanical devices. His book, "The Book of Knowledge of Ingenious Mechanical Devices," described various innovative machines, including water clocks and automated musical instruments.

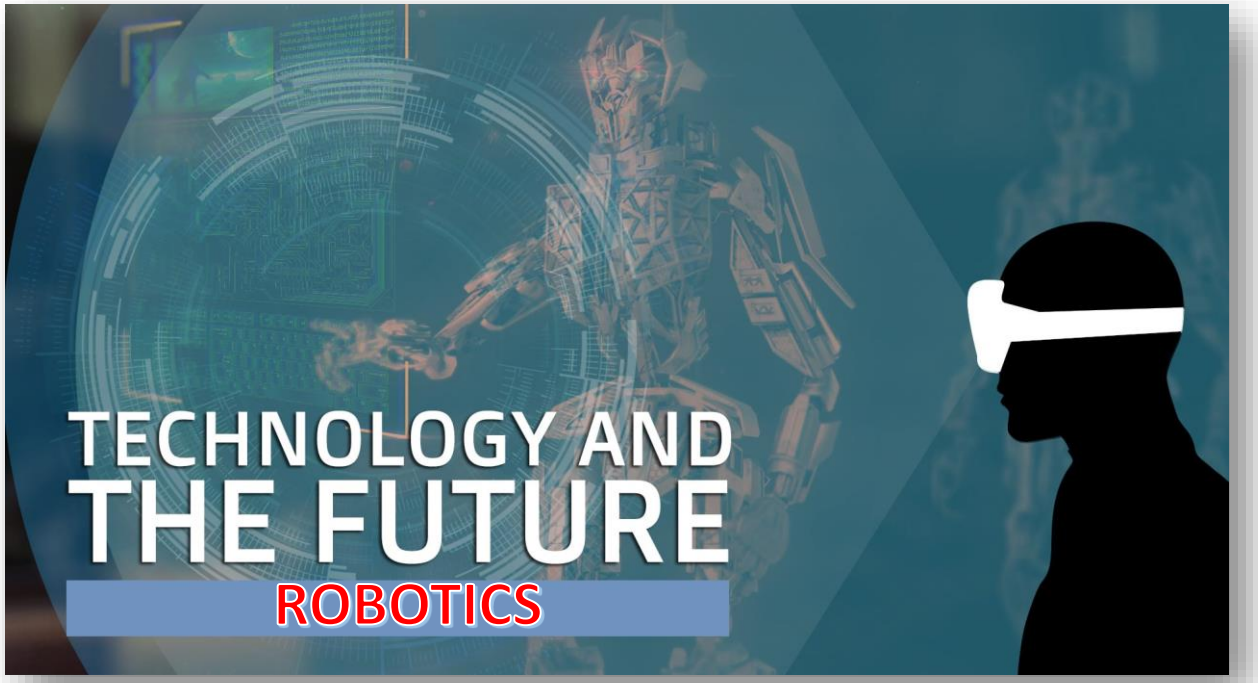
3- Isaac Newton

contributions to physics and mathematics, he is not typically associated with robotics.

Day

01

MIDOCEAN UNIVERSITY

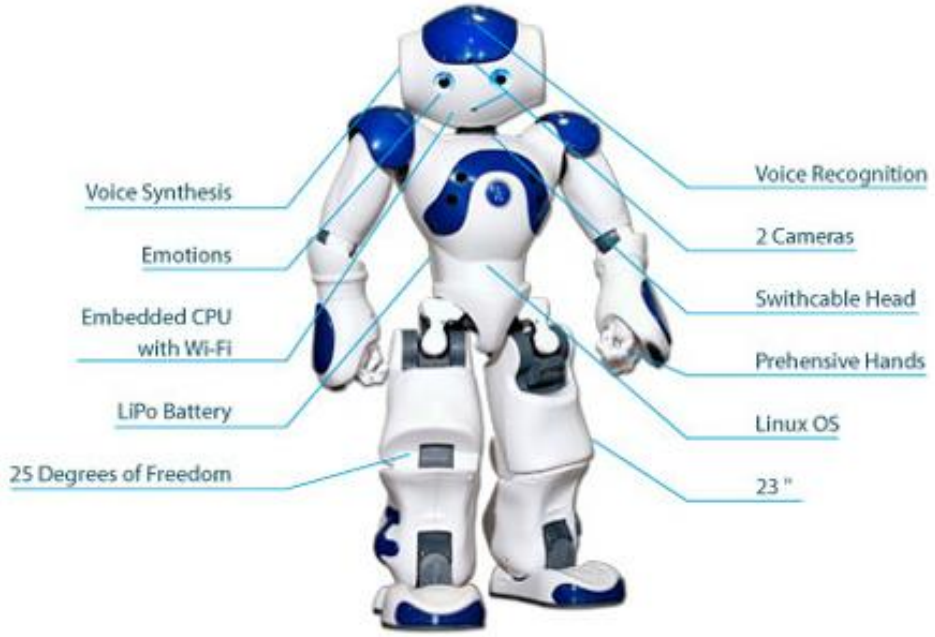


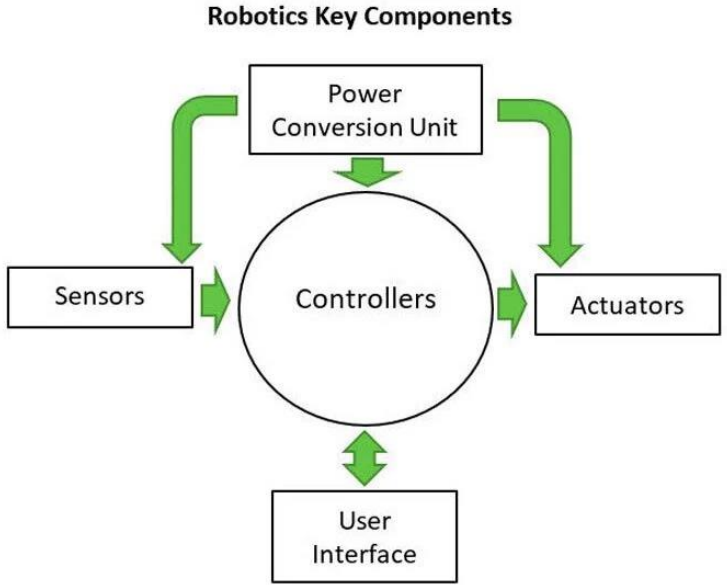
The Core of Robotics

Day
01

The Core Components of a Robot

An illustrative diagram showing the robot's core components: Computer Parts, Mechanical Components, Sensors, and Embedded Systems, leading to Human-Robot Interaction.



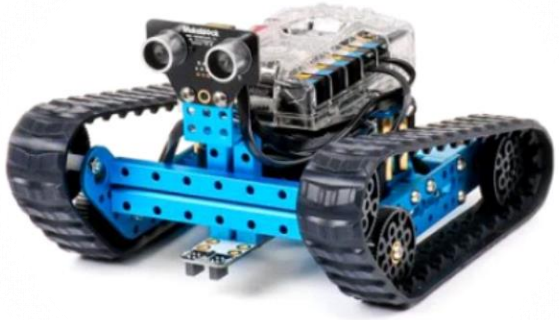


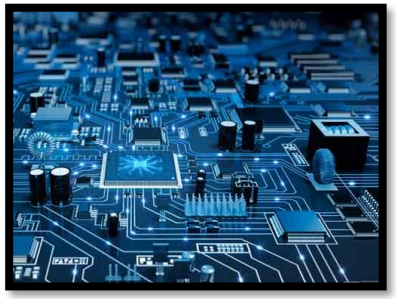
The Core of Robotics : Understanding Computer Parts

Day
01

Computer Parts in Robotics

Simplified explanations of CPUs, Memory, and Storage, using metaphors like the robot's "brain" and "memory bank."





Day
01

Actuators: The Muscles of Robots

1.What Are Actuators?

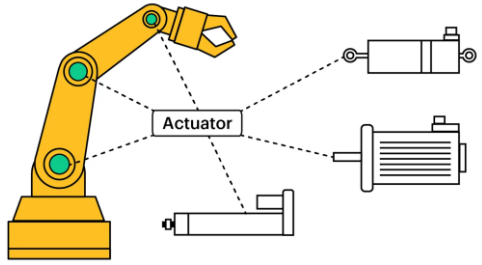
1. Brief Definition: Devices that **convert electrical energy** into **physical motion**.
2. Analogy: Just as muscles allow humans and animals to move, actuators enable robots to walk, grasp, lift, and perform other actions.

2.Types of Actuators

Electric Motors: The most common type, used in robotic arms, drones, and more. Illustrate with a snippet showing an electric motor in a robotic arm.

3. Why Actuators Are Important

Emphasize the role of actuators in expanding what robots can do, making them capable of interacting with the physical world in diverse and complex ways.



Day
01

Motors: Powering Robotic Movement

MIDOCEAN UNIVERSITY

1.Introduction to Motors in Robotics

1. Brief explanation of how motors are the driving force behind robotic movement, acting as the "muscle" that powers everything from robotic arms to autonomous vehicles.

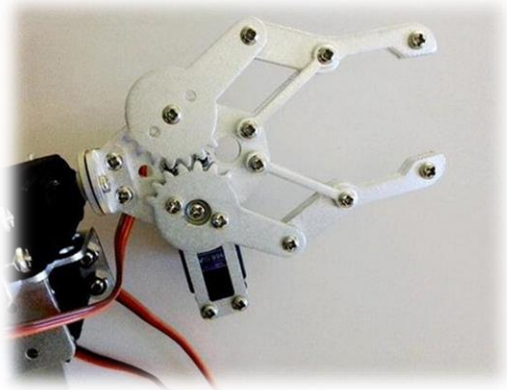
2.Types of Motors

1. **Servo Motors**:
    1. Characterized by their ability to control angular position with precision. Commonly used in robotic arms, antennas, or any application where precise positioning is required.
2. **Stepper Motors**:
    1. Known for their accuracy in controlling steps, making them ideal for applications requiring exact movements, such as 3D printers and CNC machines.
3. **DC Motors**:
    1. Direct Current motors offer simple control and are used in a wide range of applications due to their speed variability and torque. Examples include hobby robots, drones, and electric vehicles.

Day

01

Servo Motor



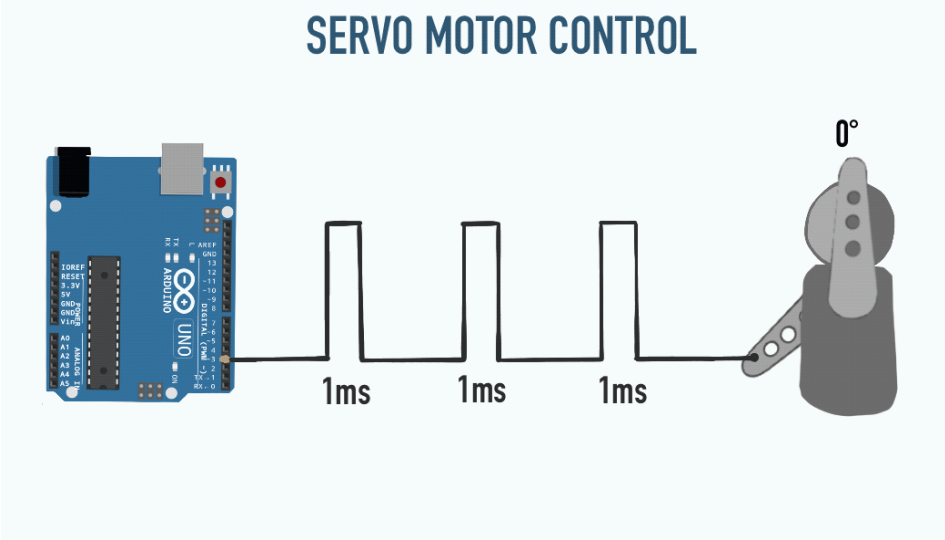
Day

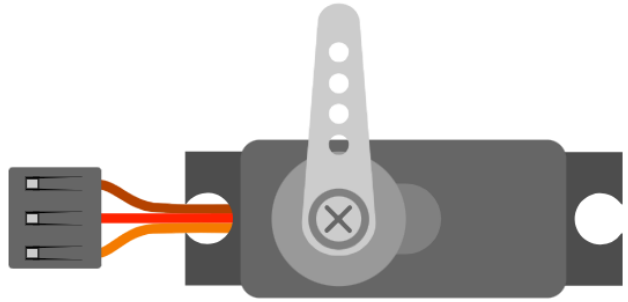
01

Servo motor

You can rotate the servo motor shaft from 0° to 180° by varying the PWM pulse width from 1 ms to 2 ms.

PWM (Pulse Width Modulation)





NAME	WIRE COLOR	DESCRIPTION
PWM	Yellow	Input control signal for the motor
V+	Red	Positive supply terminal
Ground	Brown	Ground terminal

Day

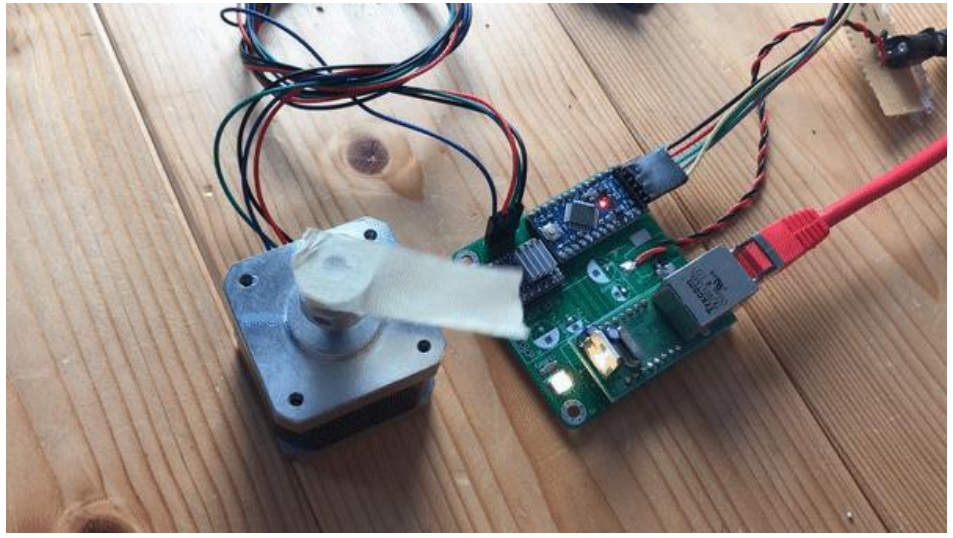
01

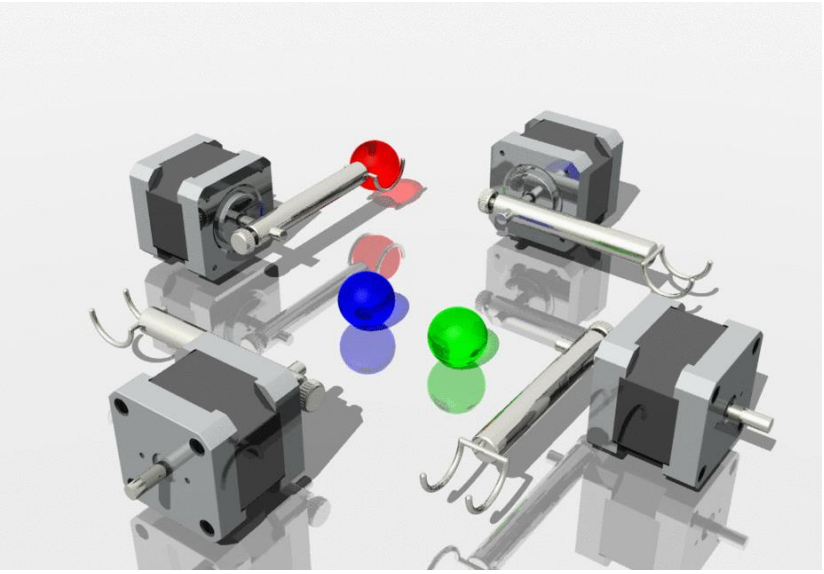
Stepper Motor

Day

01

Stepper Motor

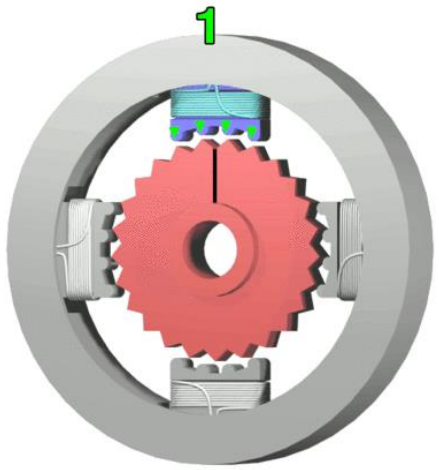


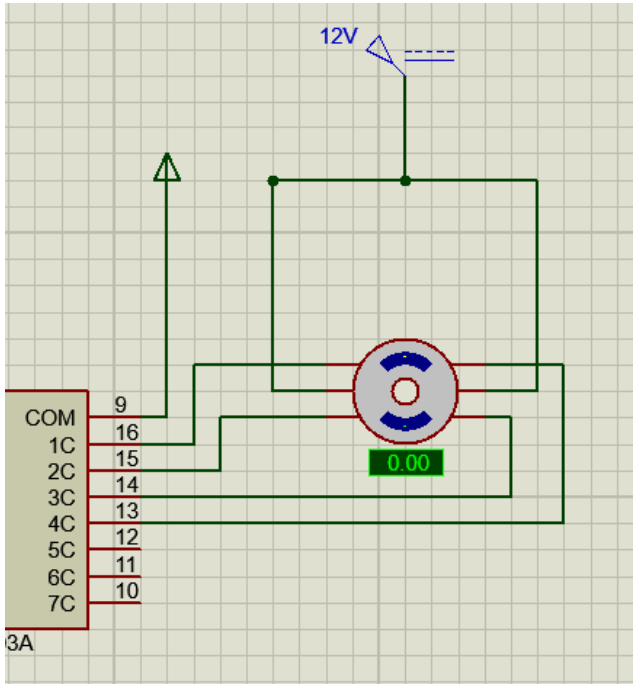


Day

01

Stepper Motor

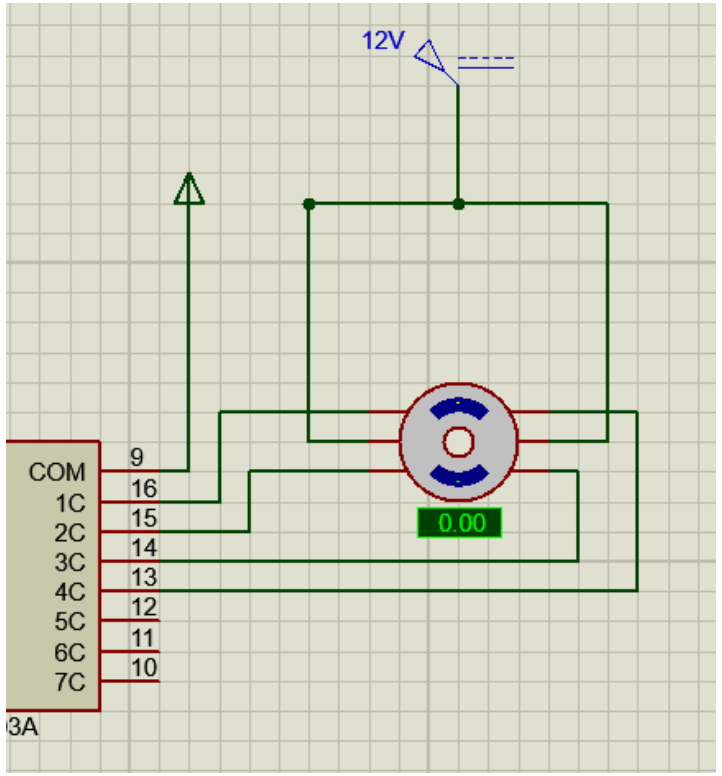




Day

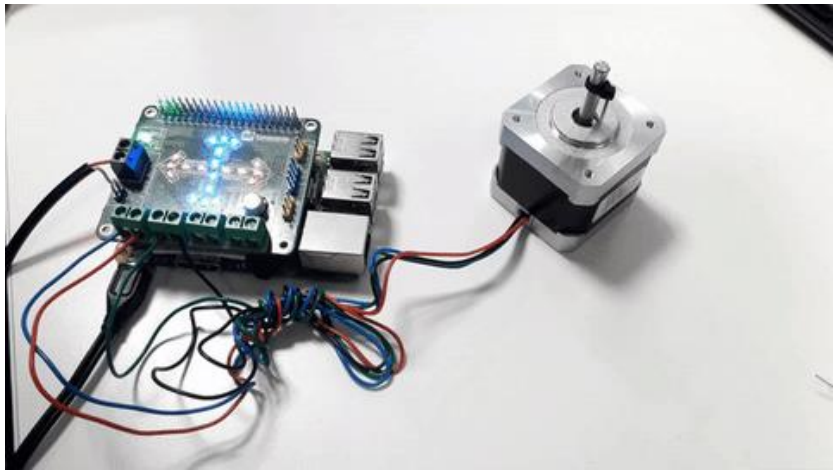
01

Stepper Motor



In the following table, A-B-C-D refers to the stator coils, that are to be energized sequentially in the manner and ‘1’s and ‘0’s refers to ‘HIGH‘ and ‘LOW’ states.

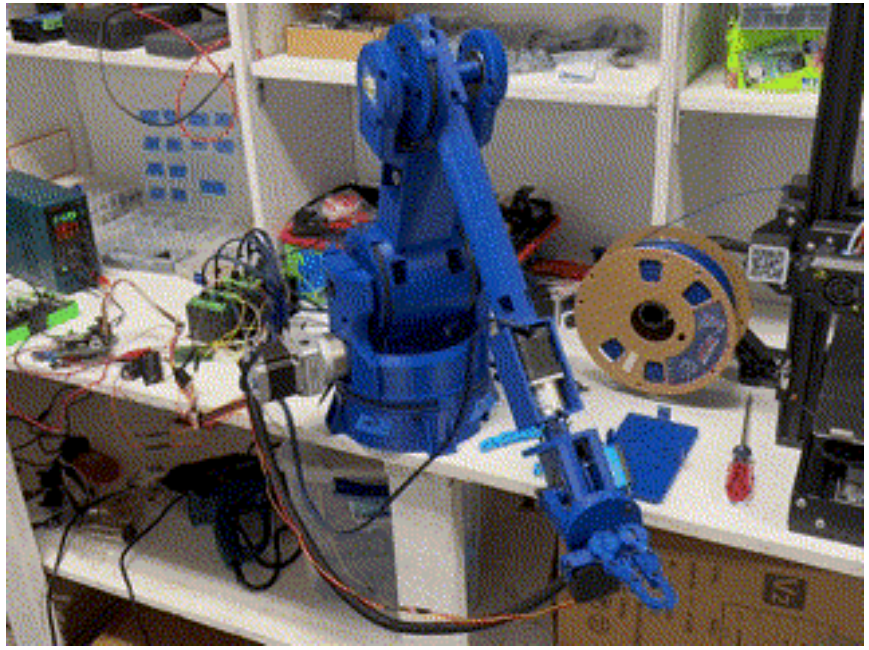
Steps	A	B	C	D	HEX
1	1	0	0	0	0x08
2	0	1	0	0	0x04
3	0	0	1	0	0x02
4	0	0	0	1	0x01

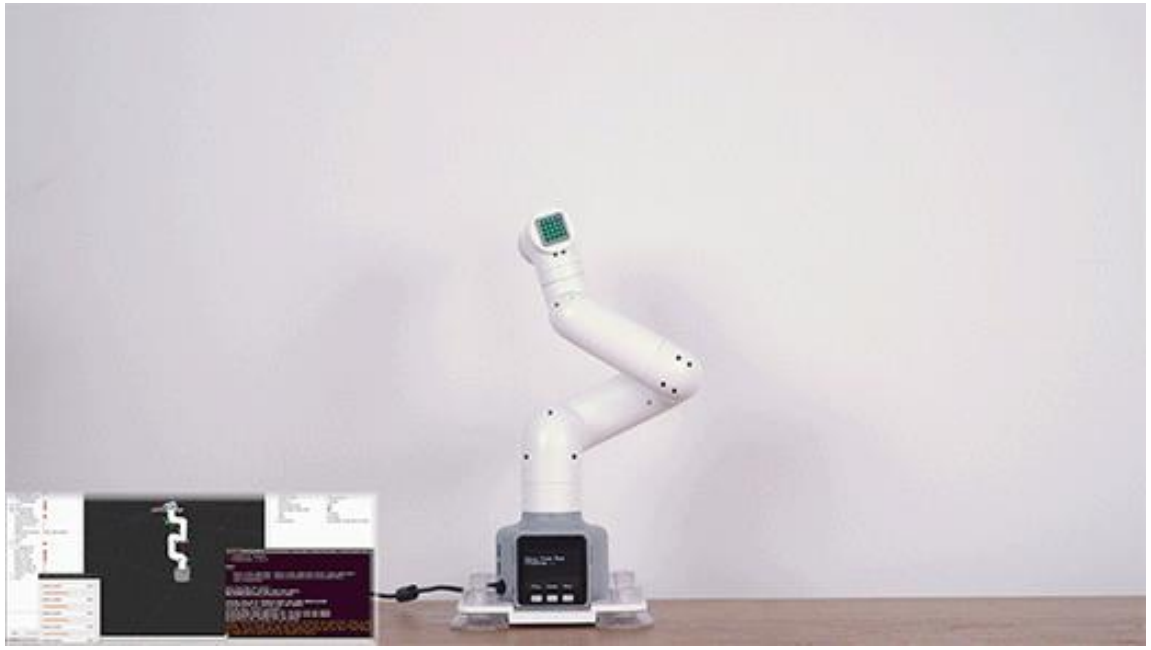


Day

01

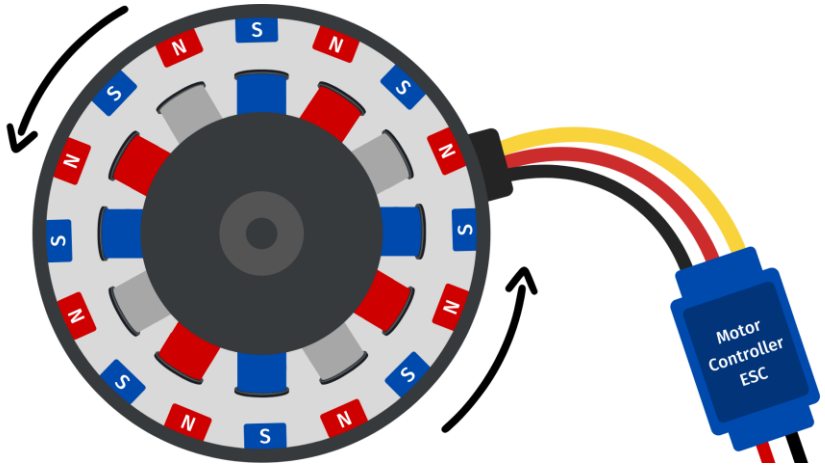
Stepper Motor





Day
01

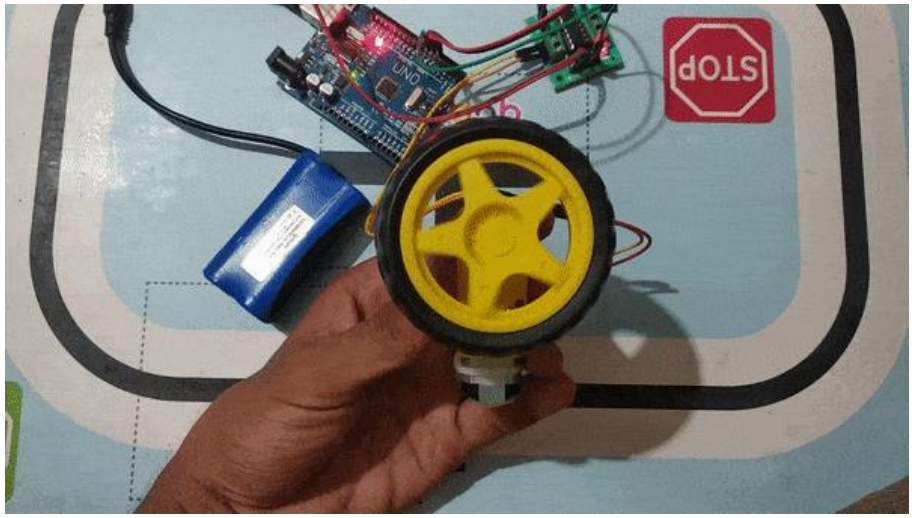
DC Motor

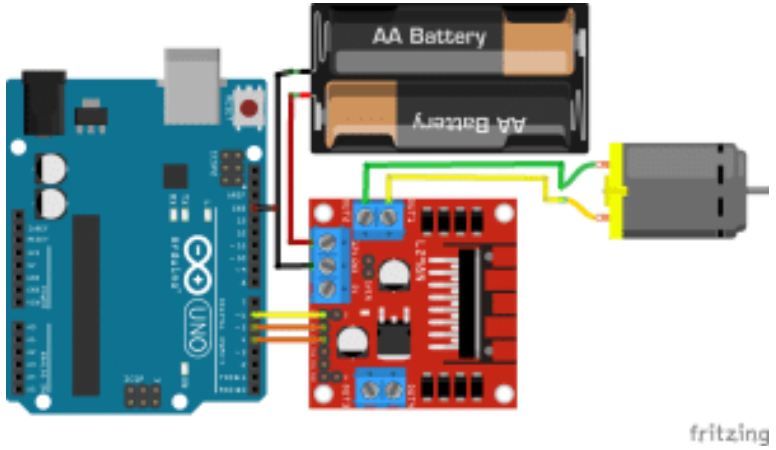


Day

01

DC Motor

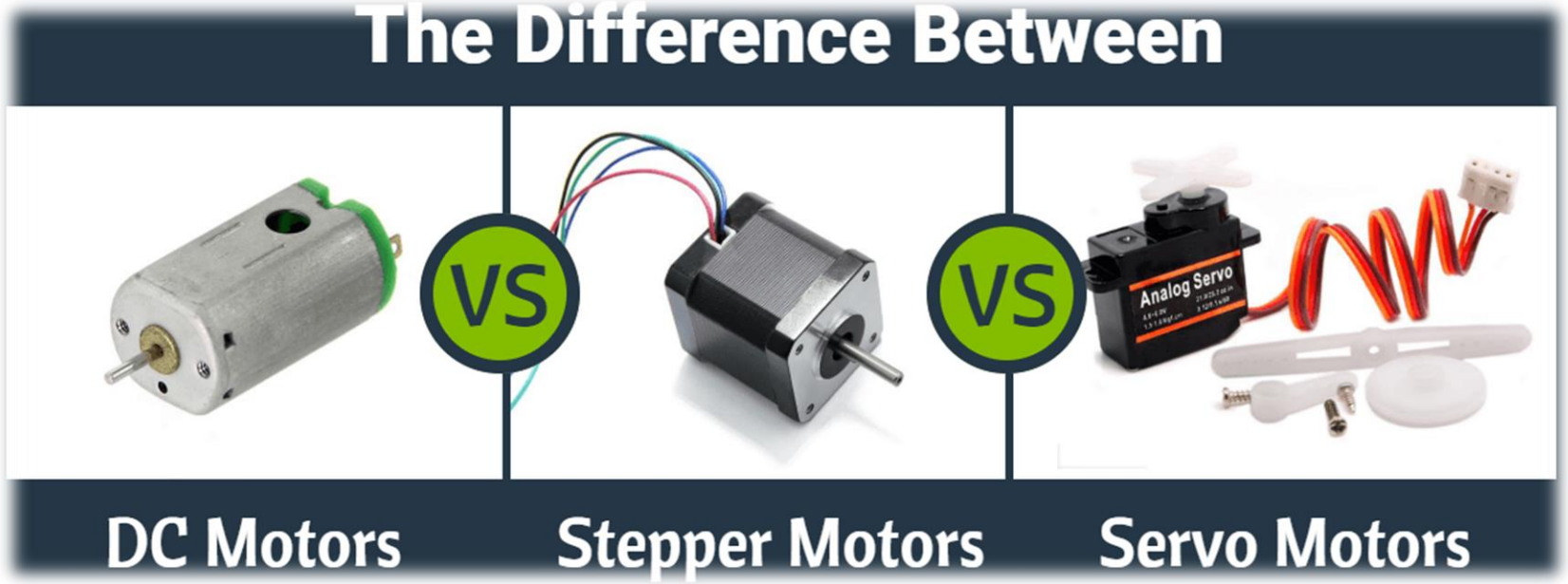




Arduino uno	Motor driver
D5	IN1
D4	IN2
GND	GND

Day
01

Motor Conclusion



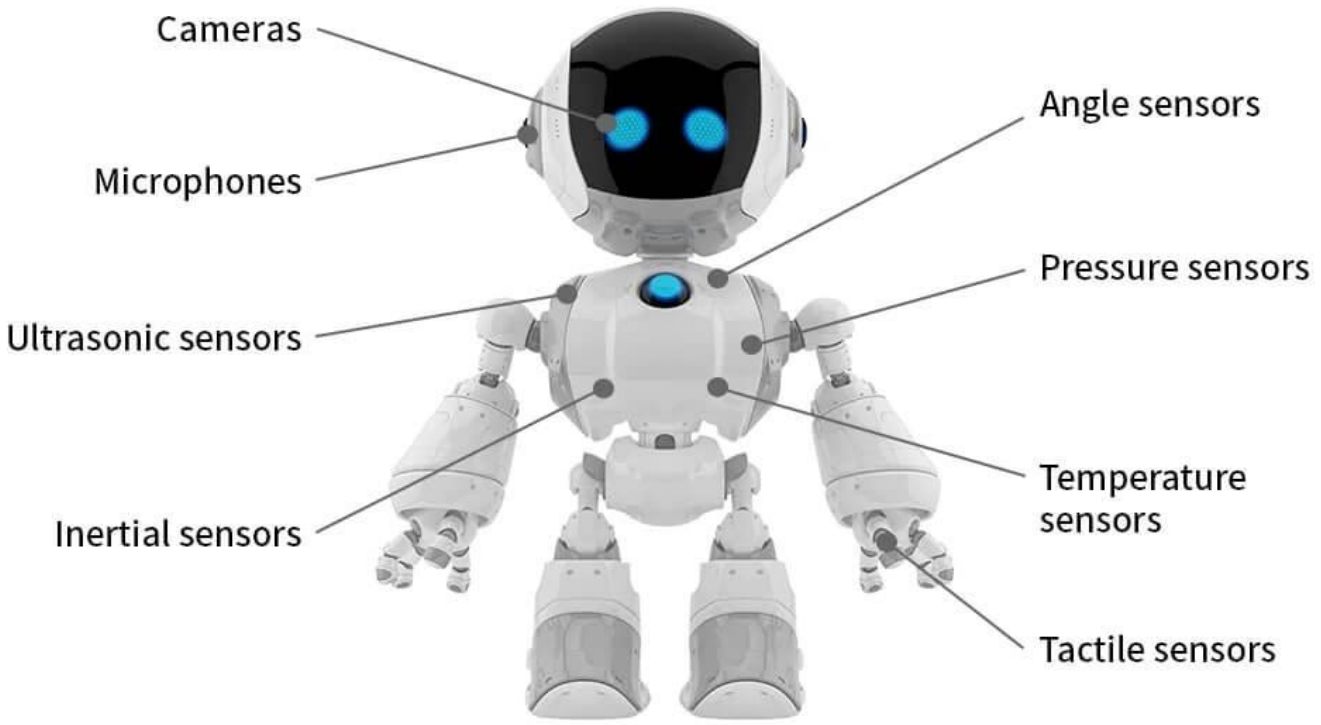
Day
01

Fundamental differences between DC, Servo, and Stepper motors

Motor Type	Control	Precision	Speed	Torque	Primary Use in Robotics
DC Motor	Variable speed control	Low	High	Medium	Ideal for applications requiring variable speed and direction, such as robot wheels or drones.
Servo Motor	Positional control	High	Medium	High at low speeds	Best for precise control of movement, such as robotic arms or where specific positioning is crucial.
Stepper Motor	Step control	High	Low to Medium	High at low speeds	Suited for precise steps in motion, making them perfect for 3D printers, CNC machines, and robots requiring detailed positioning without feedback.

Day
01

Sensors



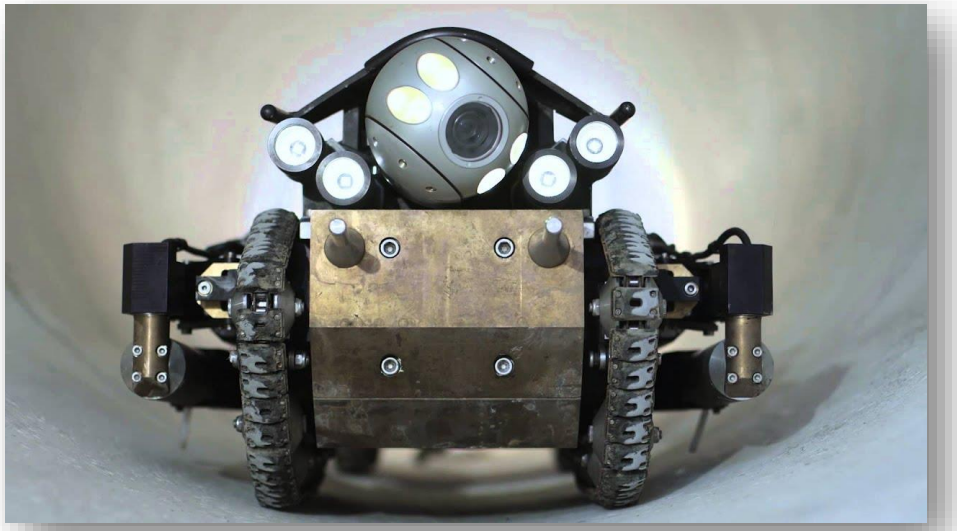
Vision Sensors: The Eyes of Robots

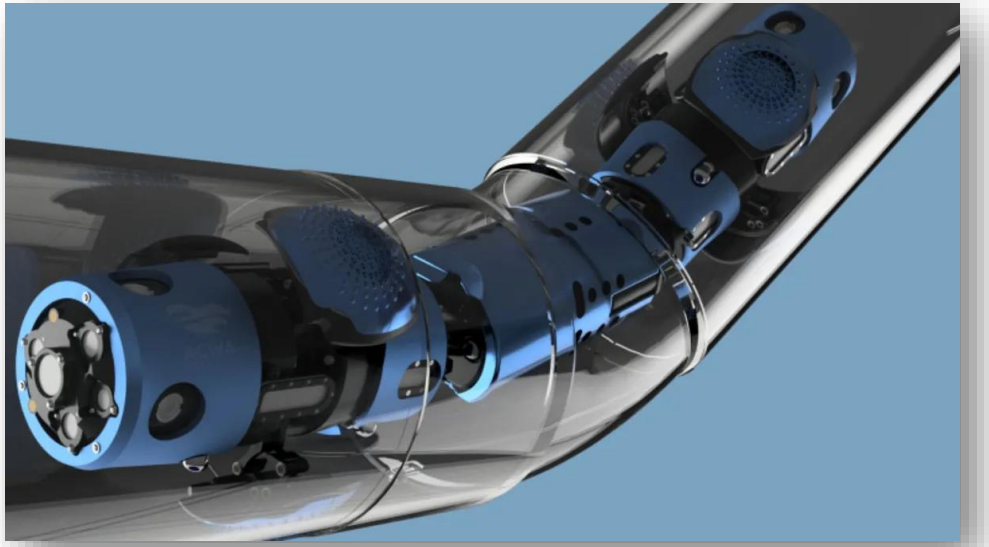
- **Overview**: Introduction to how vision sensors allow robots to perceive their environment visually, simulate human sight.
- **Types**:
  - **Cameras (2D and 3D)**: Capture visual information. 2D cameras provide flat images, while 3D cameras add depth perception, crucial for tasks requiring spatial understanding.
  - **LIDAR (Light Detection and Ranging)**: Uses laser pulses to map out the robot's surroundings in high detail, essential for navigation and obstacle avoidance.
- **Applications**:
  - Object recognition, navigation, and environmental mapping.

Day

01

2D Camera Sensor

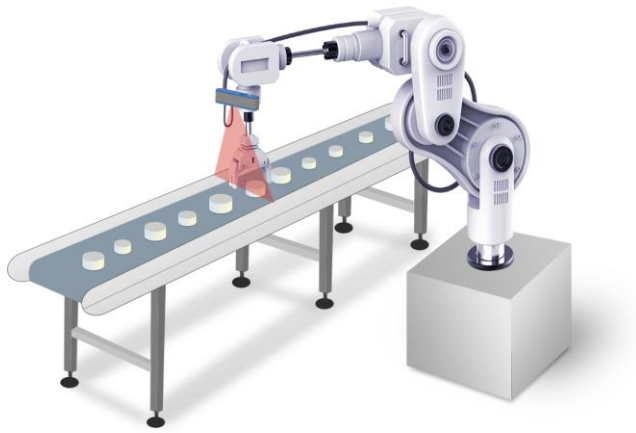




Day

01

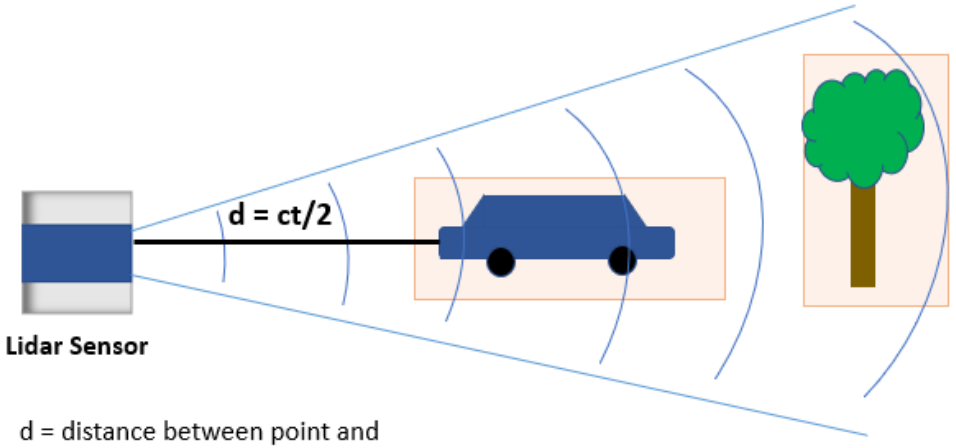
3D Camera Sensor



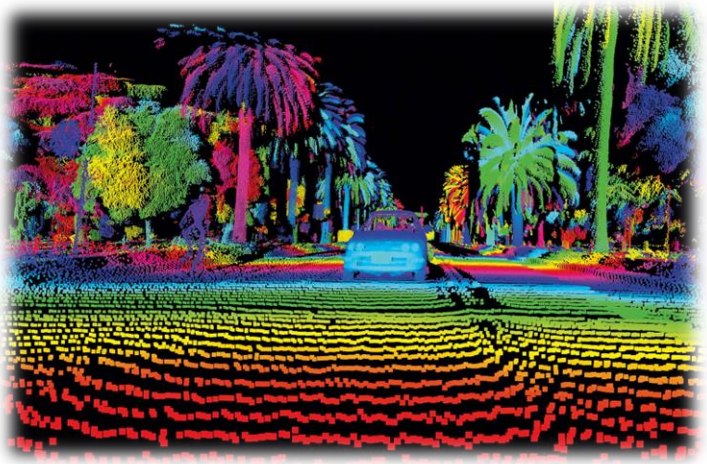
Day

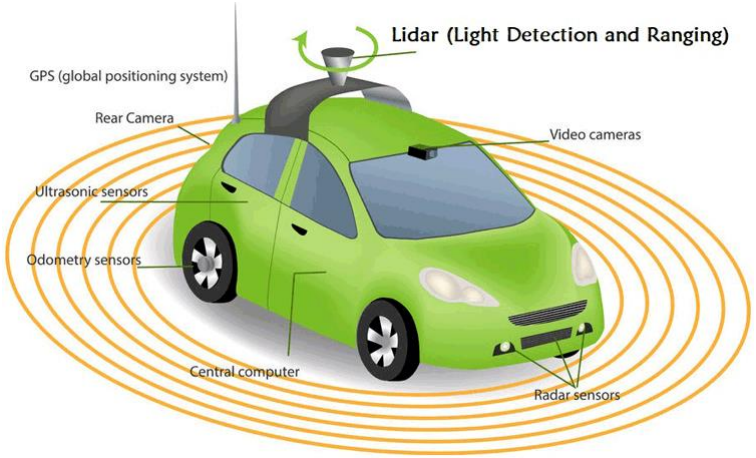
01

LIDAR (Light Detection and Ranging):



d = distance between point and the sensor
c = speed of light
t = time of flight





Proximity and Distance Sensors

Navigating Space: Proximity and Distance Sensors

•**Overview**: Explaining the role of proximity and distance sensors in detecting the presence of objects and measuring the distance to them.

•**Types**:

- **Ultrasonic Sensors**: Emit ultrasonic waves and measure the reflection time to determine distance to objects. Widely used in robotic cars for obstacle detection.
- **Infrared Sensors**: Use infrared light to detect objects and estimate distance, useful in line-following robots and simple navigation tasks.

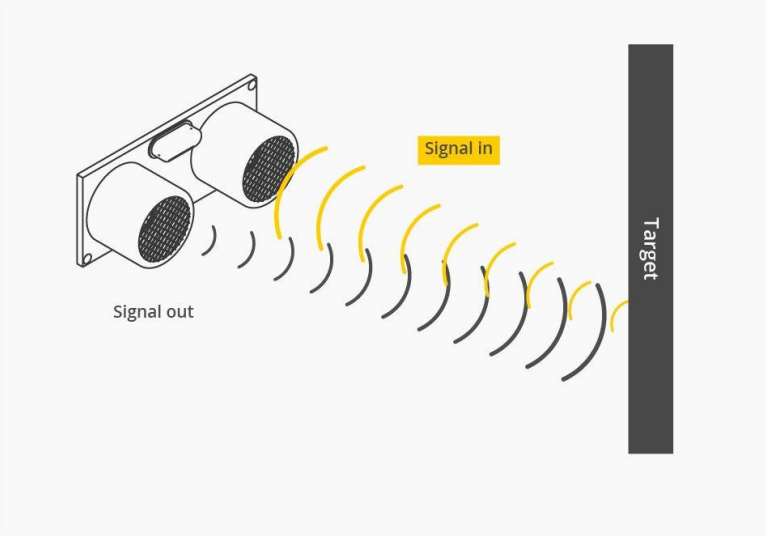
•**Applications**:

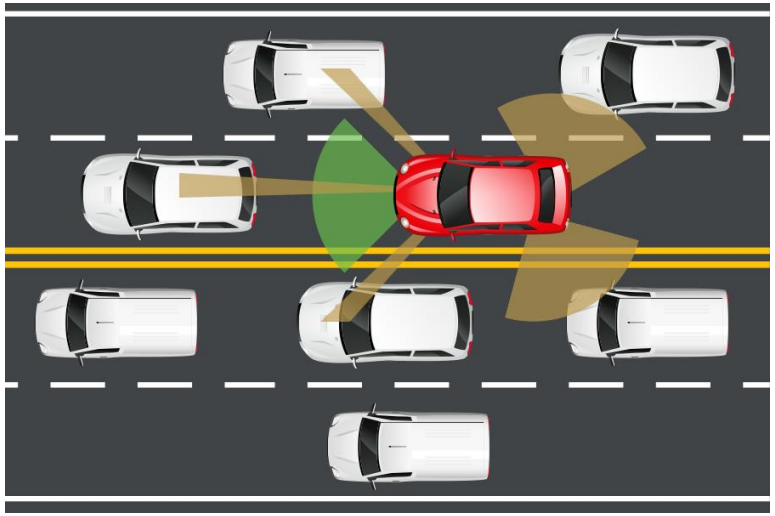
- Collision avoidance, area scanning, and precise movements in tight spaces.

Day

01

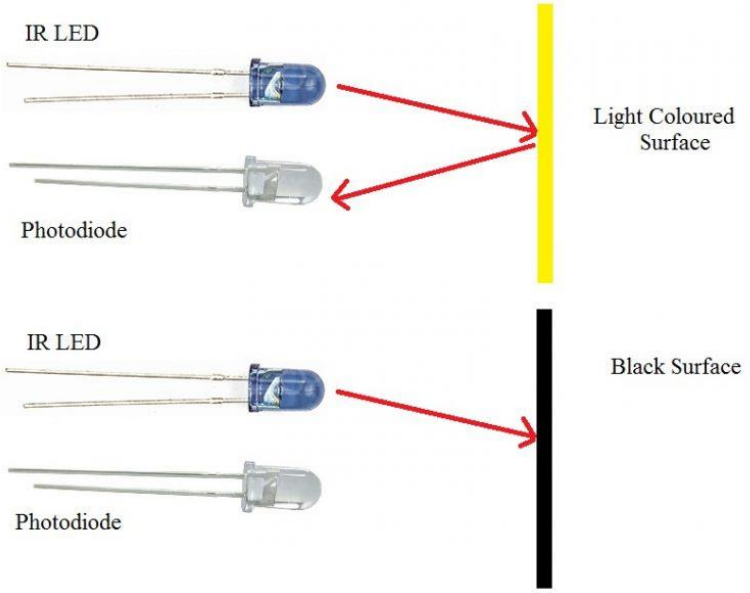
Ultra Sonic Sensor

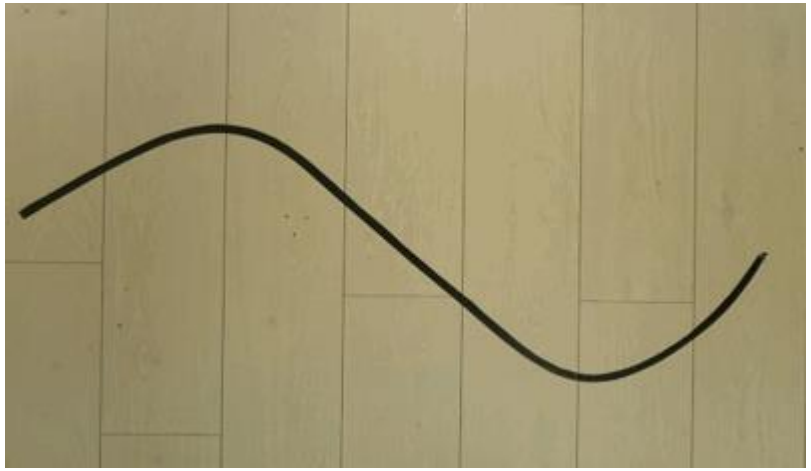




Day
01

Infra red Sensor





Feeling and Interacting: Tactile and Environmental Sensors

MIDOCEAN UNIVERSITY

•**Overview**: Highlighting sensors that allow robots to 'feel' their environment and respond to physical interactions or changes in their surroundings.

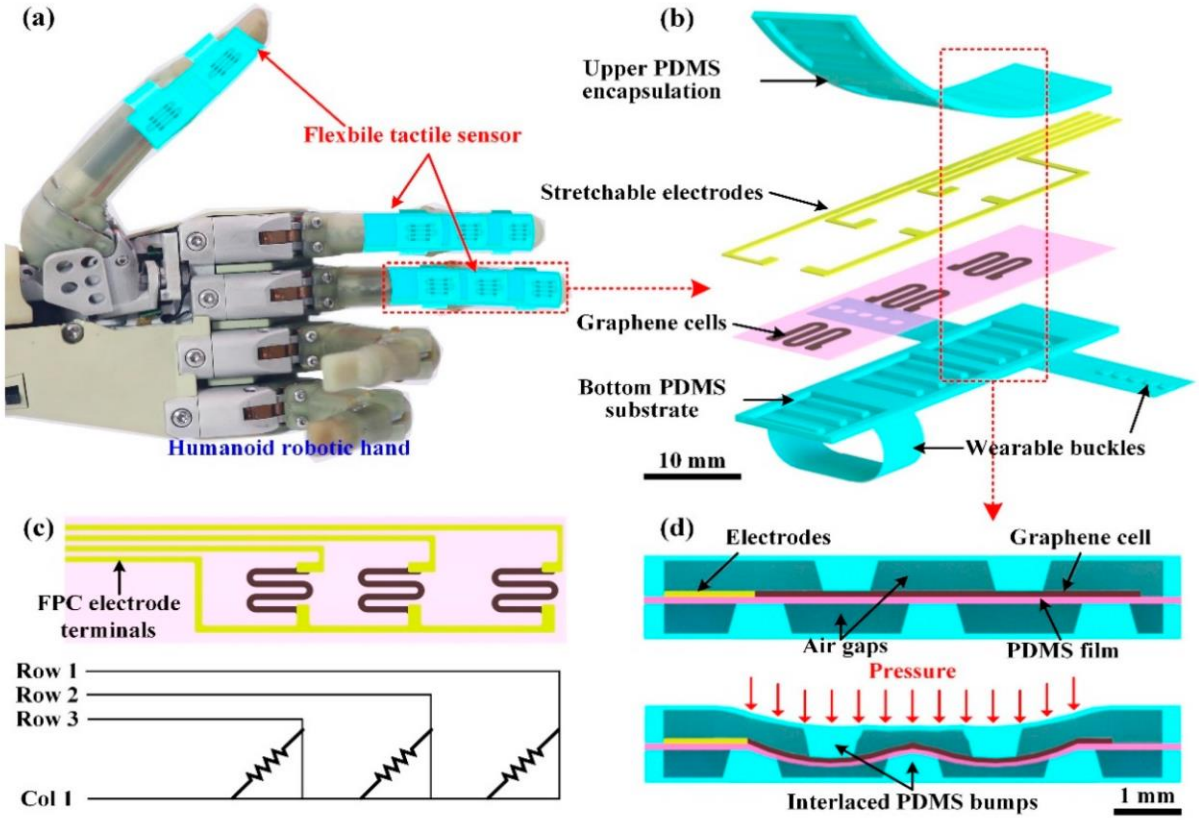
•**Types**:

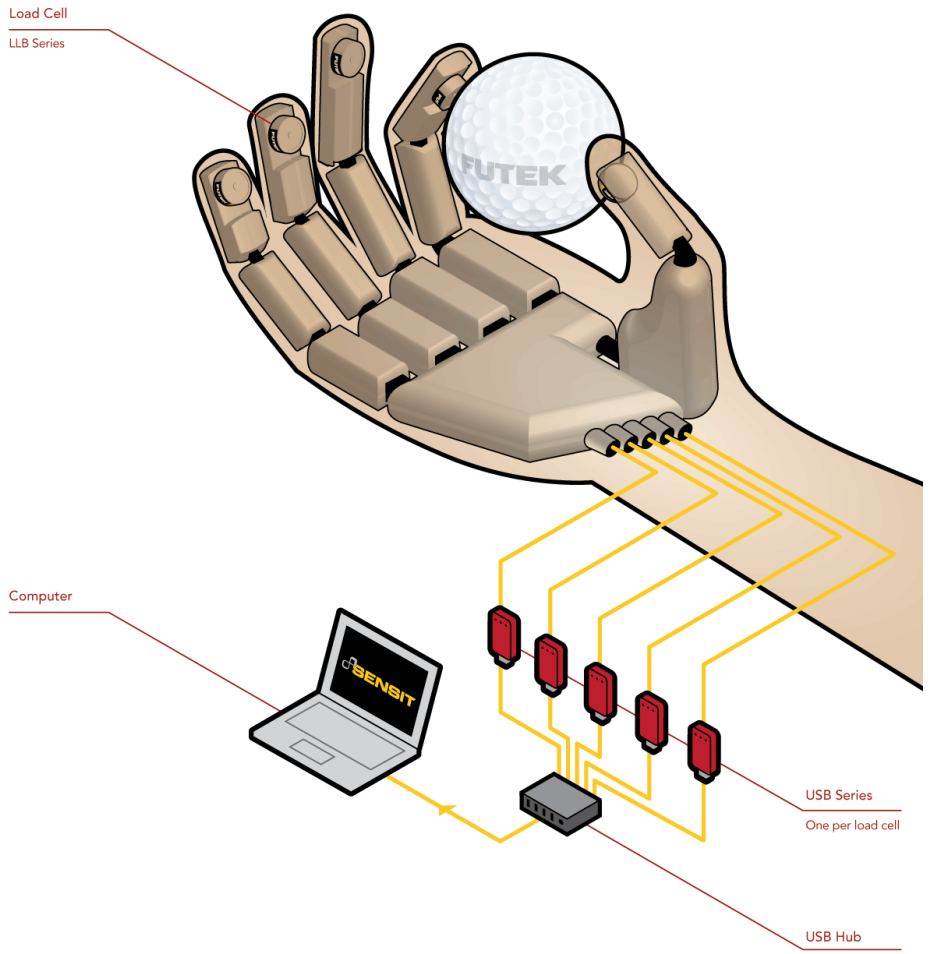
- **Tactile Sensors**: Enable robots to detect physical contact, pressure, and textures. Key in applications requiring delicate handling or manipulation, like robotic hands.
- **Environmental Sensors**: Measure temperature, humidity, or gas concentrations, critical for robots operating in varying or extreme conditions, such as disaster response robots.

•**Applications**:

- Hazard detection, material handling, and adapting to environmental conditions.

Tactile Sensors





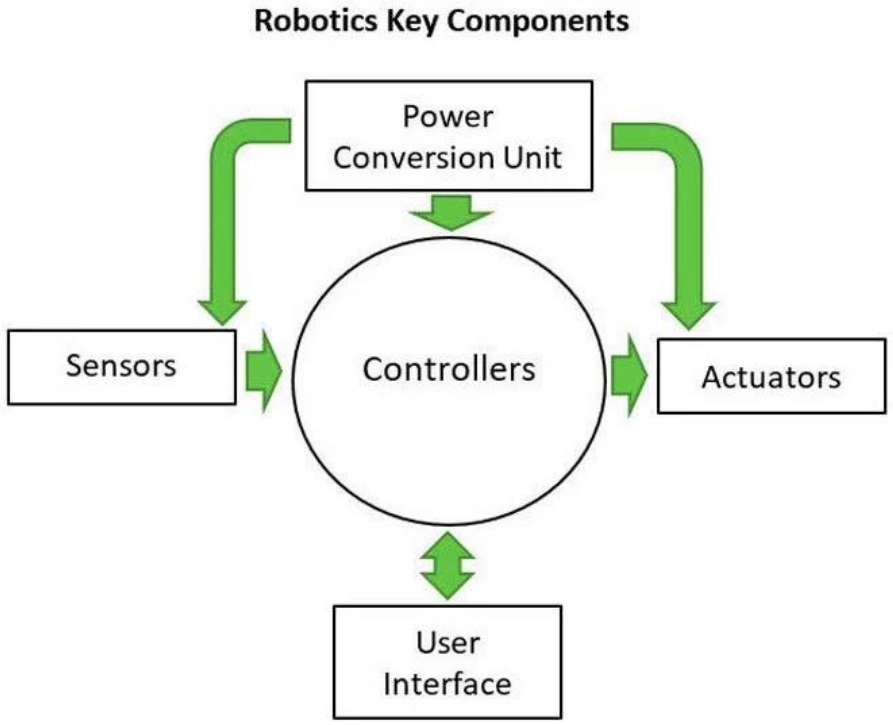
Day
01

Environmental Sensors

MIDOCEAN UNIVERSITY

Day
01

Bringing It All Together



Day
01

Introduction to Webots

MIDOCEAN UNIVERSITY

Webots

robot simulation

Webots is an open source and multi-platform desktop application used to **simulate** robots

https://cyberbotics.com/

Webots Open Source

Day

01

Introduction to Webots

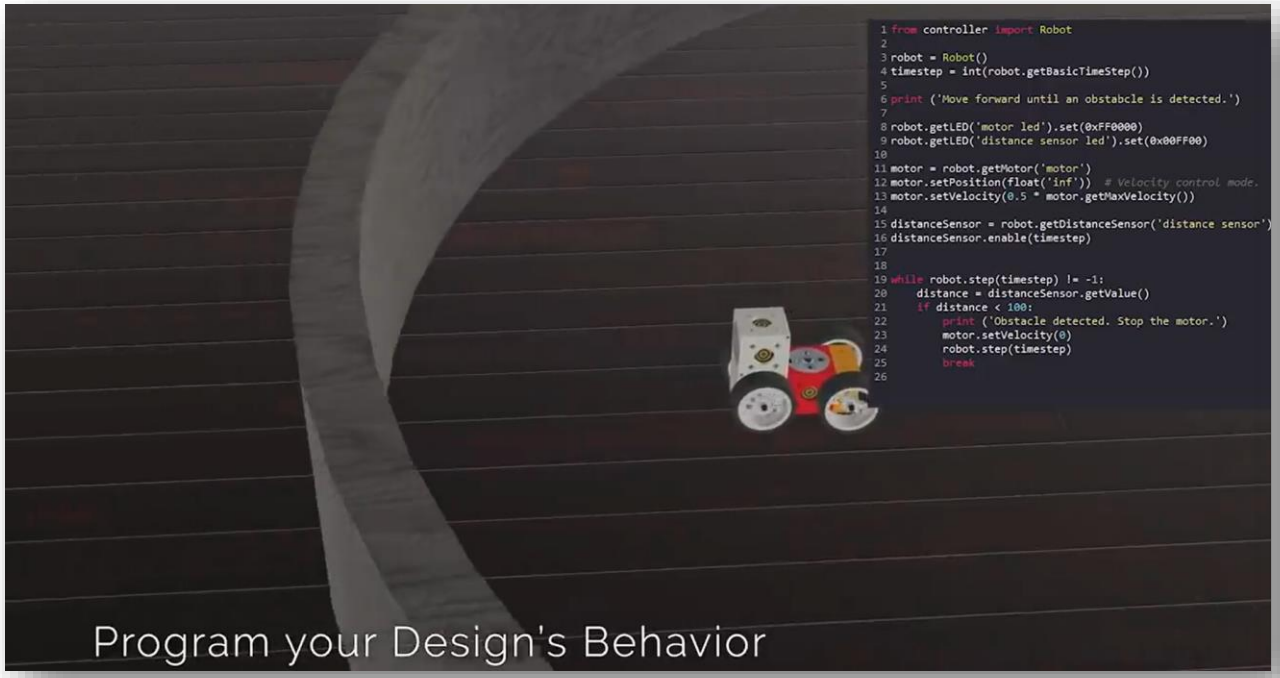
MIDOCEAN UNIVERSITY

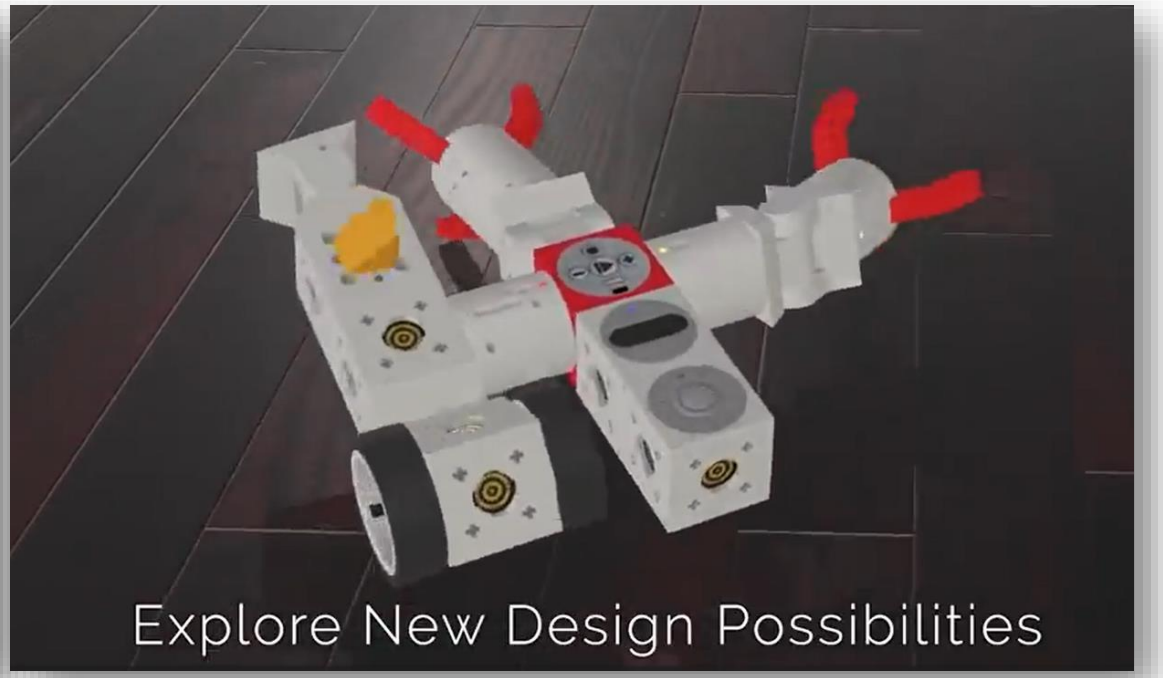
Day

01

Introduction to Webots

MIDOCEAN UNIVERSITY

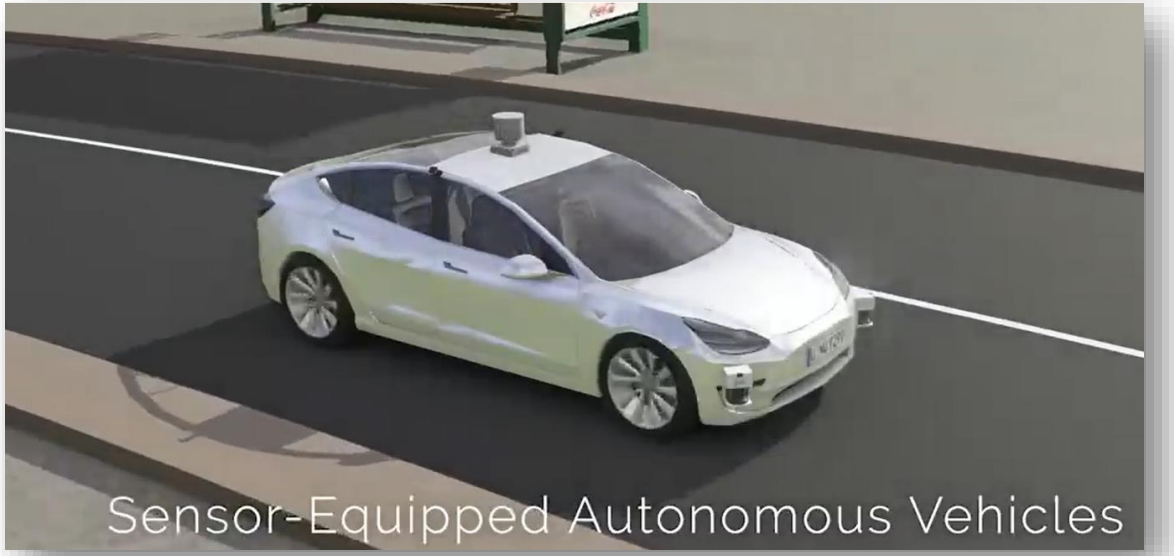




Day
01

Introduction to Webots

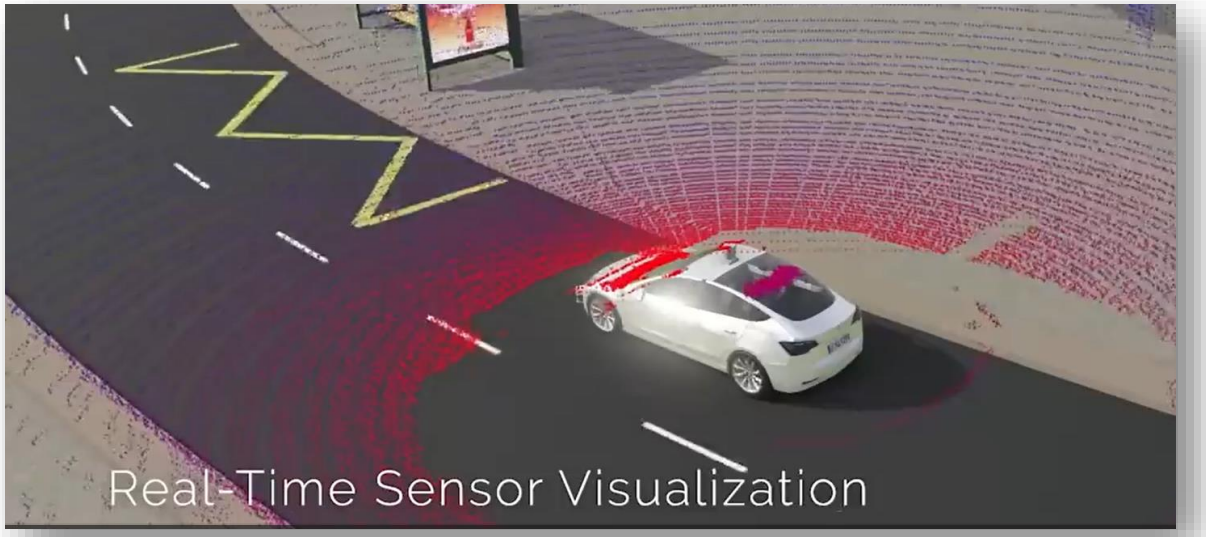
MIDOCEAN UNIVERSITY

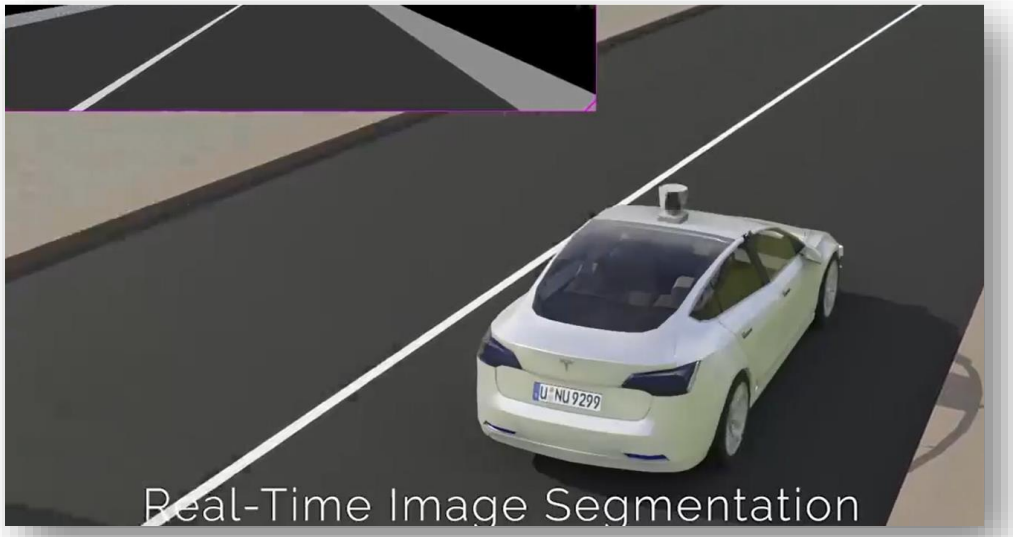


Day
01

Introduction to Webots

MIDOCEAN UNIVERSITY





Day

01

Webots is Amazing Because:

MIDOCEAN UNIVERSITY

•**Real Robots on Your Screen**: Your virtual robots in Webots move just like real ones. It’s like having a robot pal in your computer!
•**Create Your Own Robot Places**: Make all sorts of places for your robots to explore, like cities, parks, or space stations!
•**Loads of Ready-Made Robots**: Choose from many robots that are already built in Webots. Start playing and learning right away.
•**Works on Any Computer**: You can use Webots on Windows, Mac, or Linux. It's super flexible!
•**Easy Robot Coding with Python**: Use Python, a simple programming language, to tell your robots what to do. Perfect for beginners and experts!
•**Teach Robots to Think**: Make your robots smart with Python. They can learn to solve problems by themselves, like real AI!
•**Robots with Cool Parts**: Add neat parts to your robots, like cameras, hands, or sensors, so they can see, touch, and feel.
•**All-in-One Robot Kit**: Webots has everything you need to create and control amazing robots.
•**For School, Research, and Business**: You can play with Webots alone, use it in school, or even in big research projects and businesses.
•**Used by Top Universities and Companies**: Many of the best universities and big companies in America use Webots. Places like MIT, Stanford, and tech giants in Silicon Valley trust Webots for their robot projects.

Day
01

WeBots

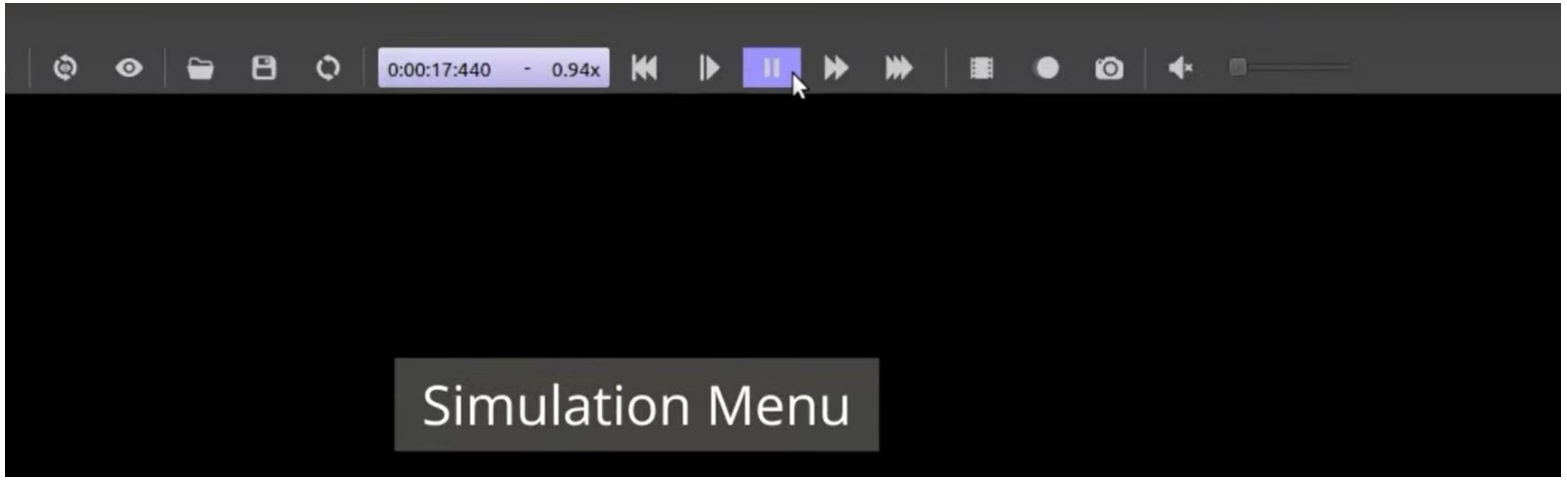
MIDOCEAN UNIVERSITY

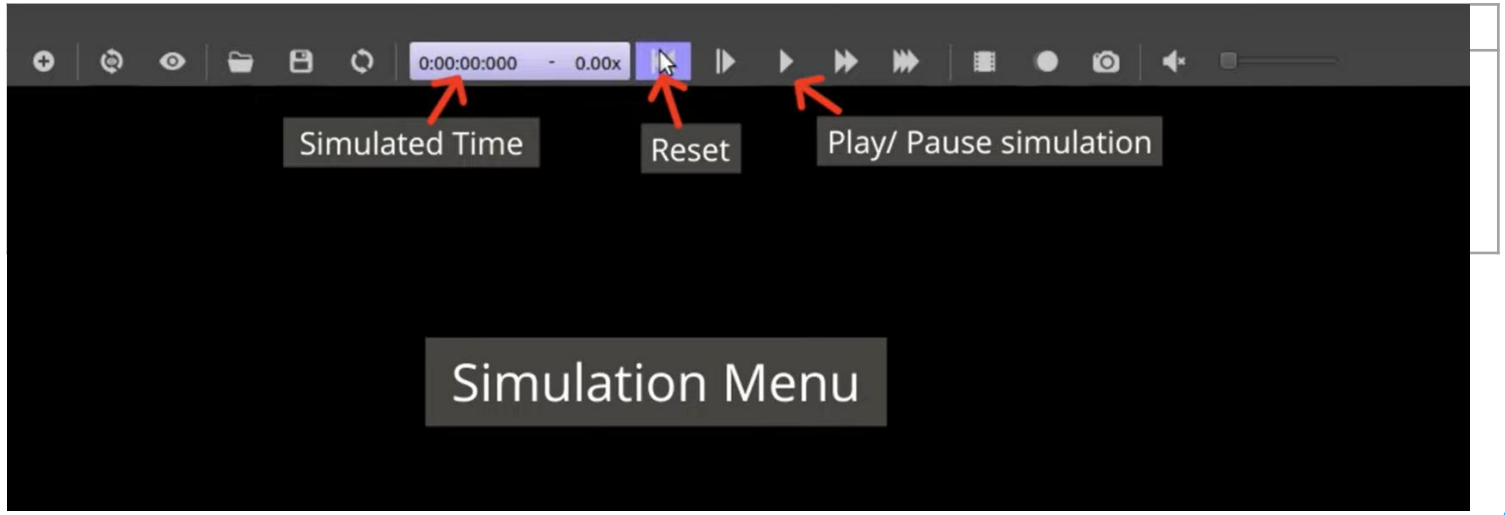
Download & Install
User interface (UI)
Create your first project
Make your own object
Understand fundamentals
Run your first simulation
Add a robot & run simulation
Where is the code

Objective
User Interface (UI)
Create your first Project

Day
01

User Interface (UI)

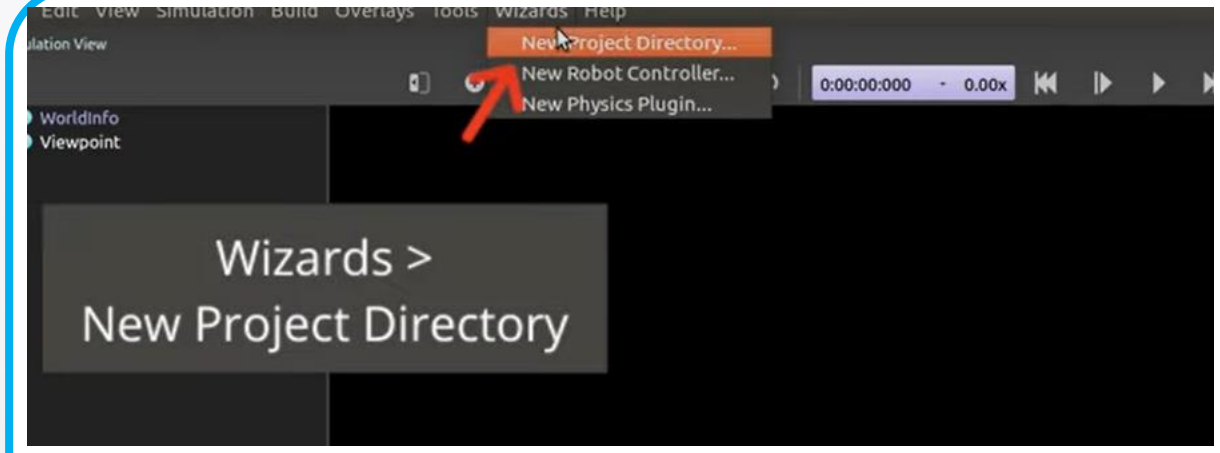


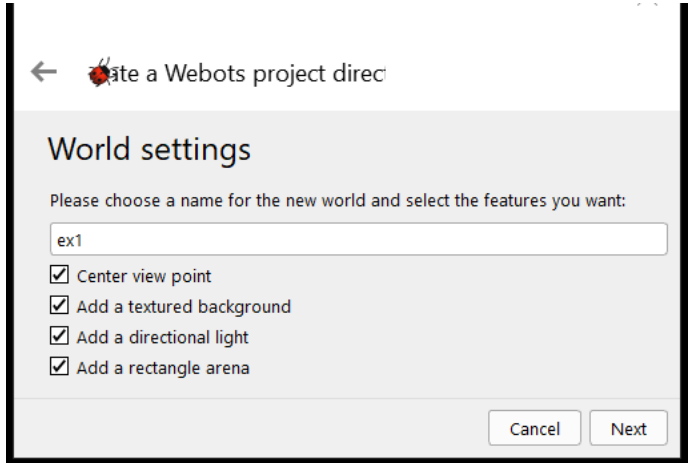


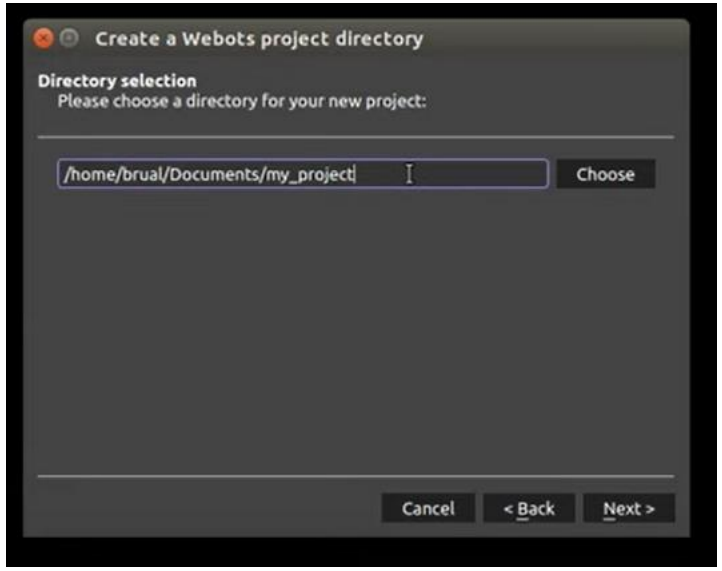
Day

01

Create new Project





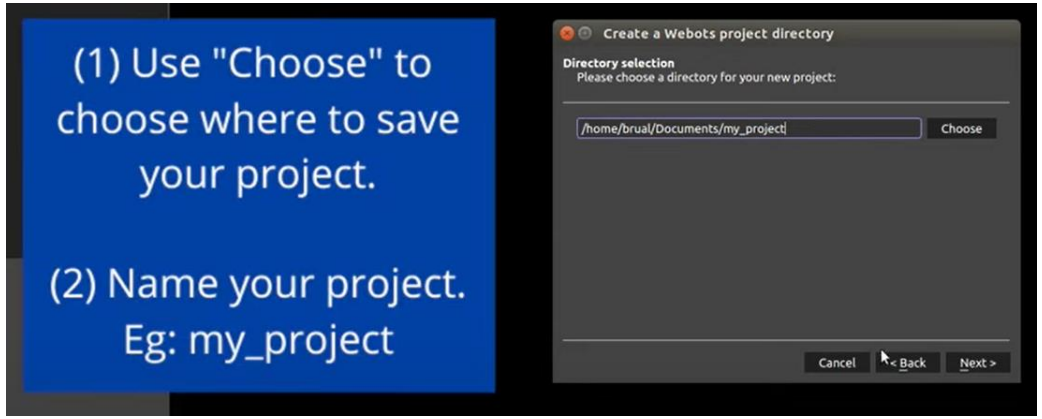


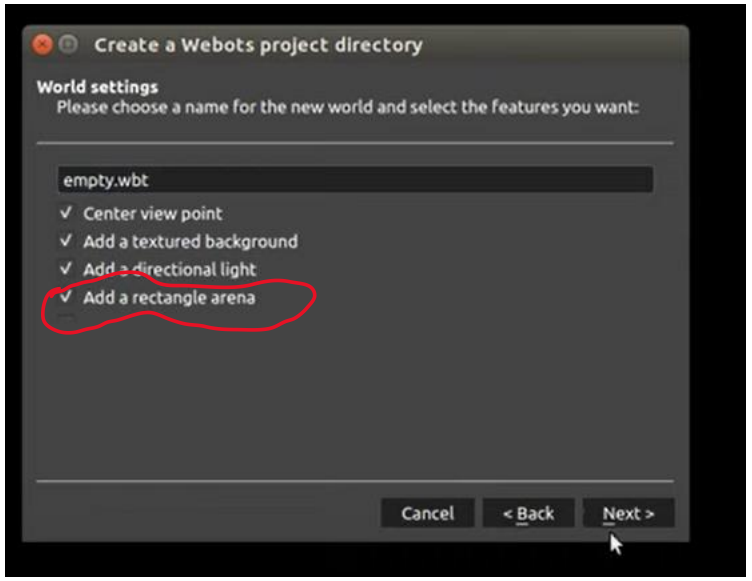
Day

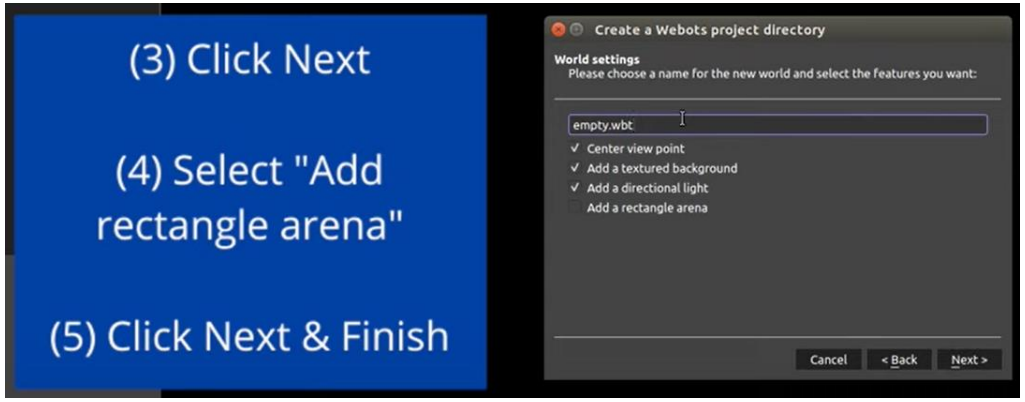
01

Create new Project

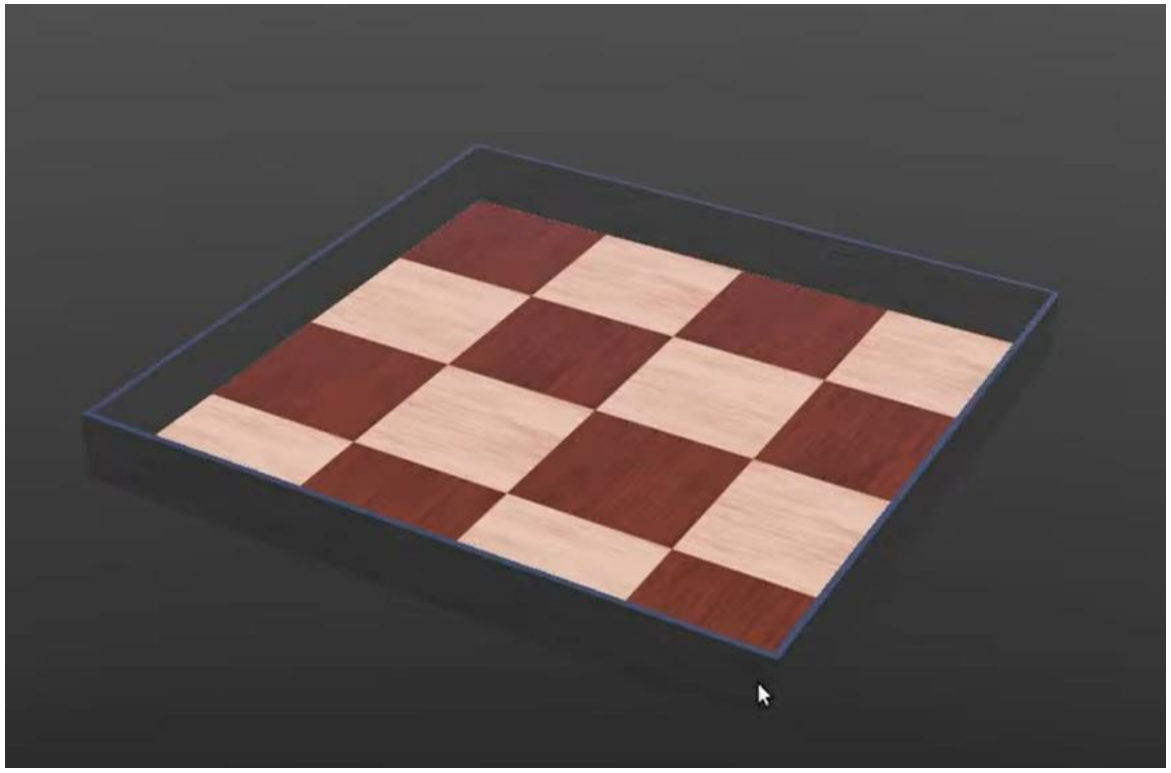
MIDOCEAN UNIVERSITY







Create new Project

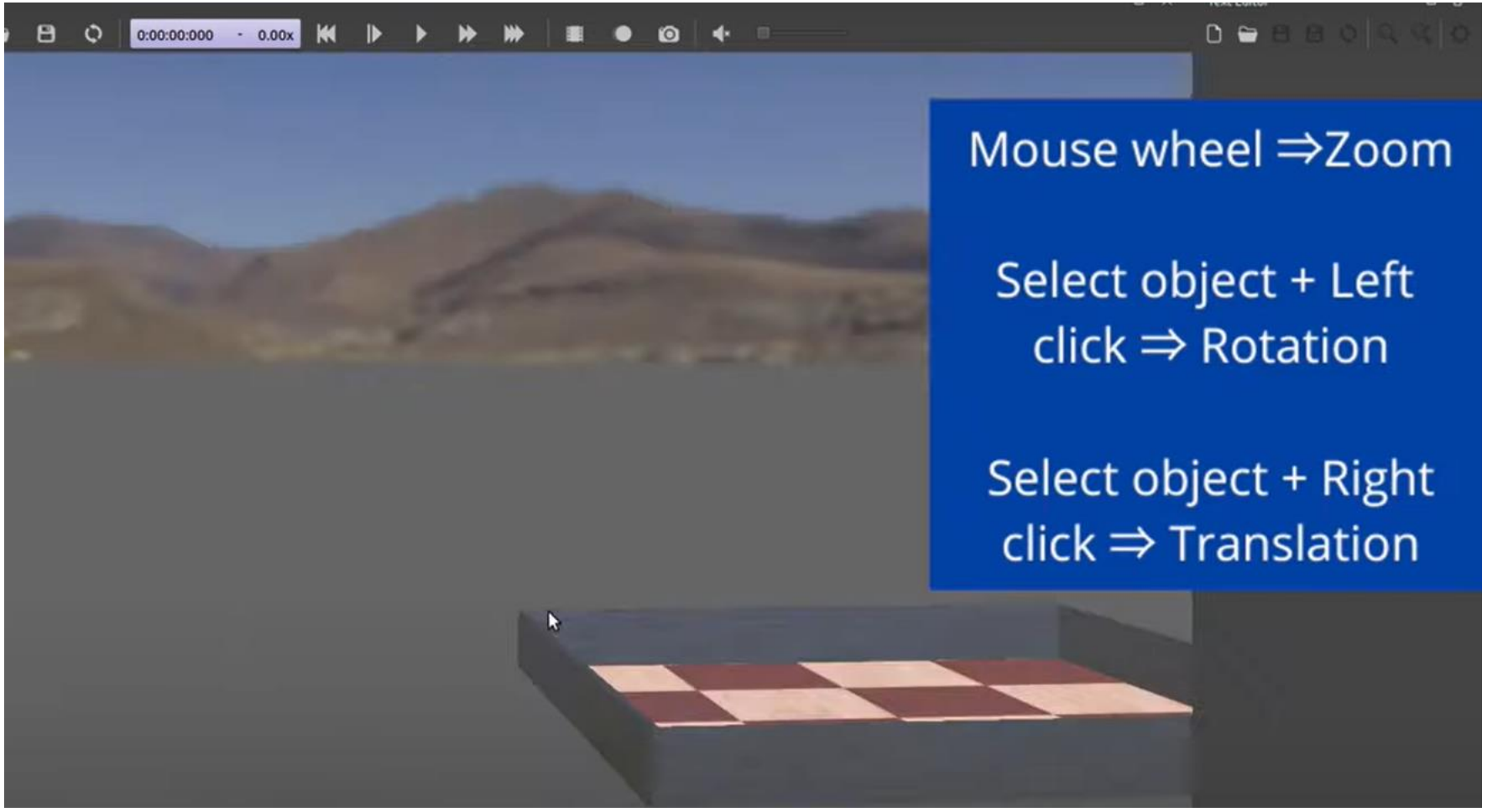


Loaded with a rectangular arena essentially a floor with four walls.

Day
01

Create new Project

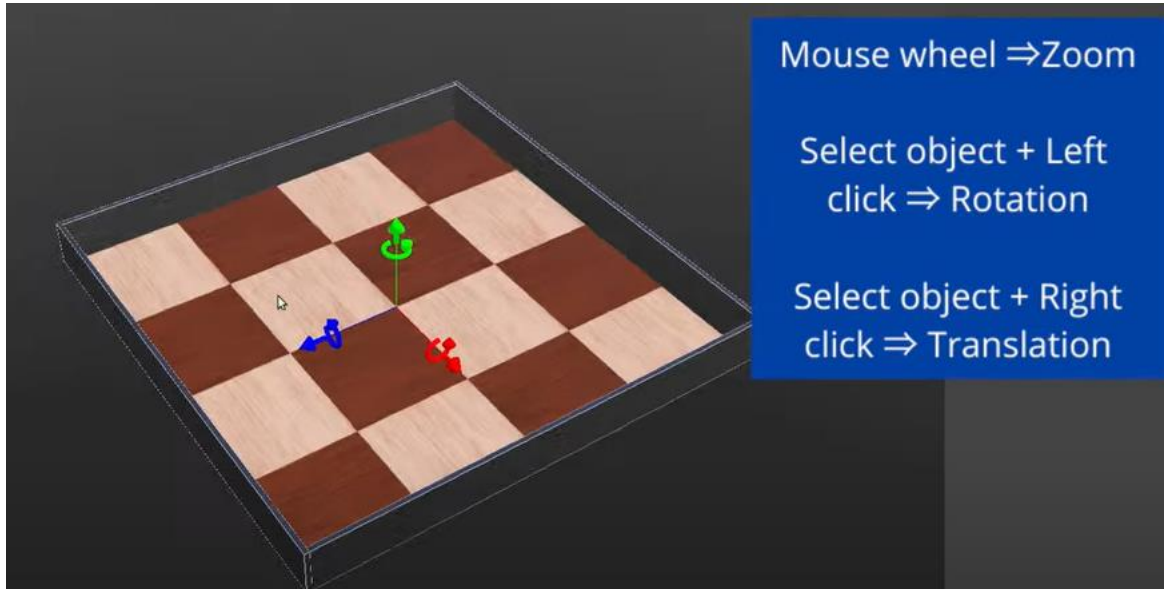
MIDOCEAN UNIVERSITY



learn how to zoom in and zoom out this is simply done by scrolling in and out

Day
01

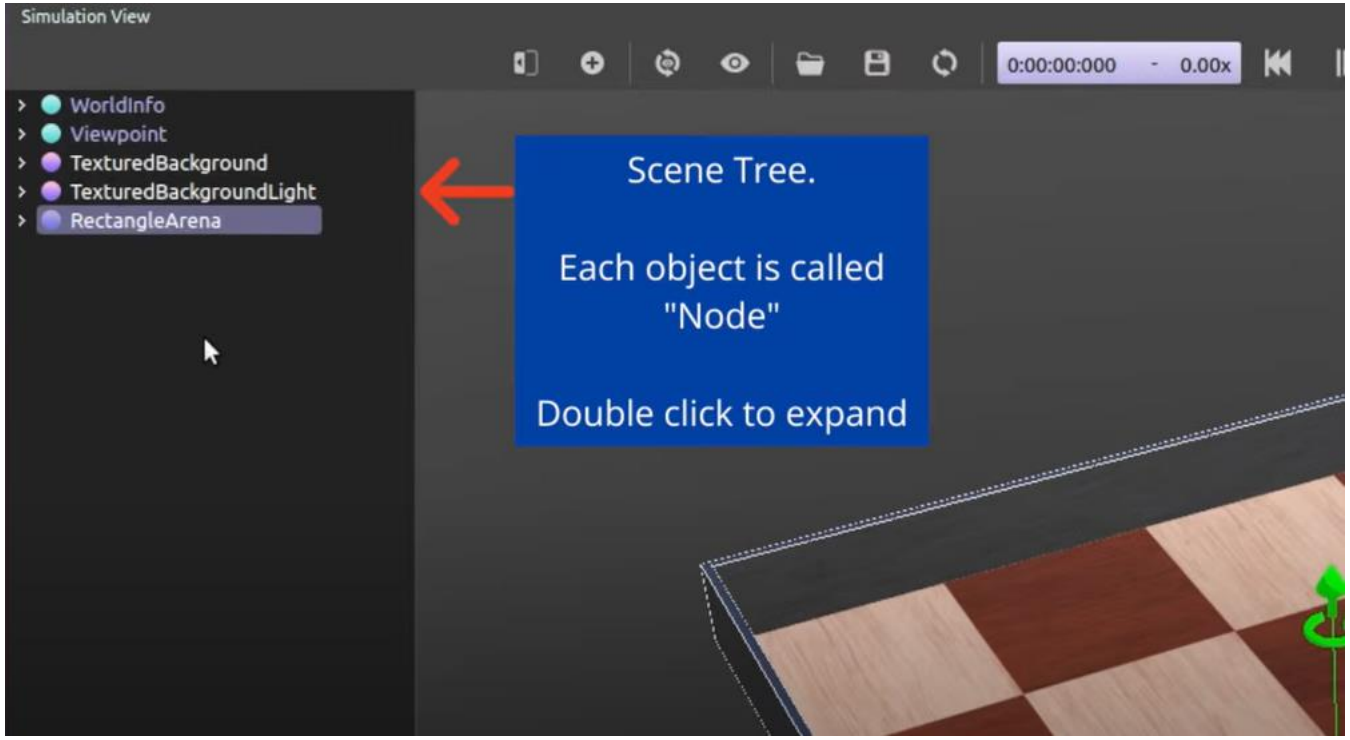
Create new Project



you can click on it and you can move it around to look at different views.

Day
01

Create new Project

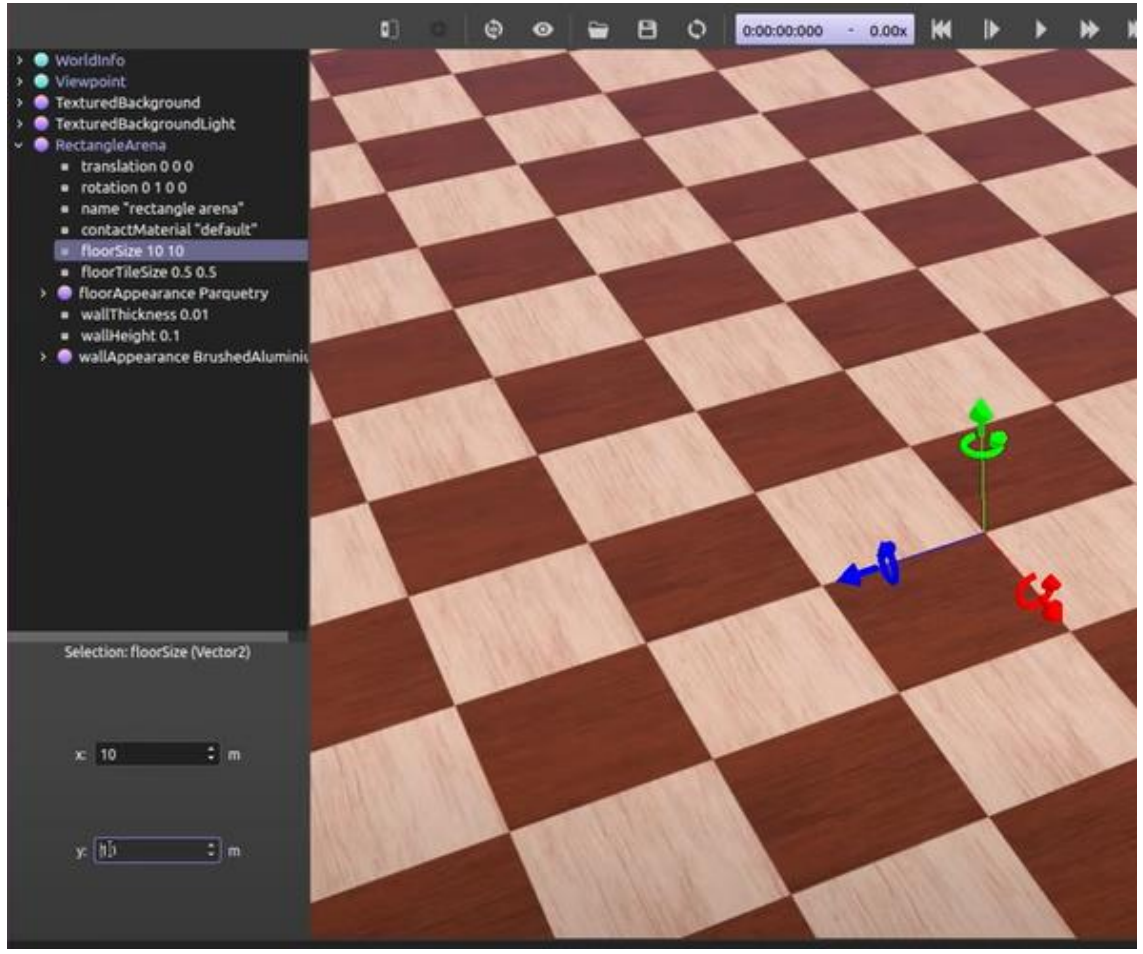


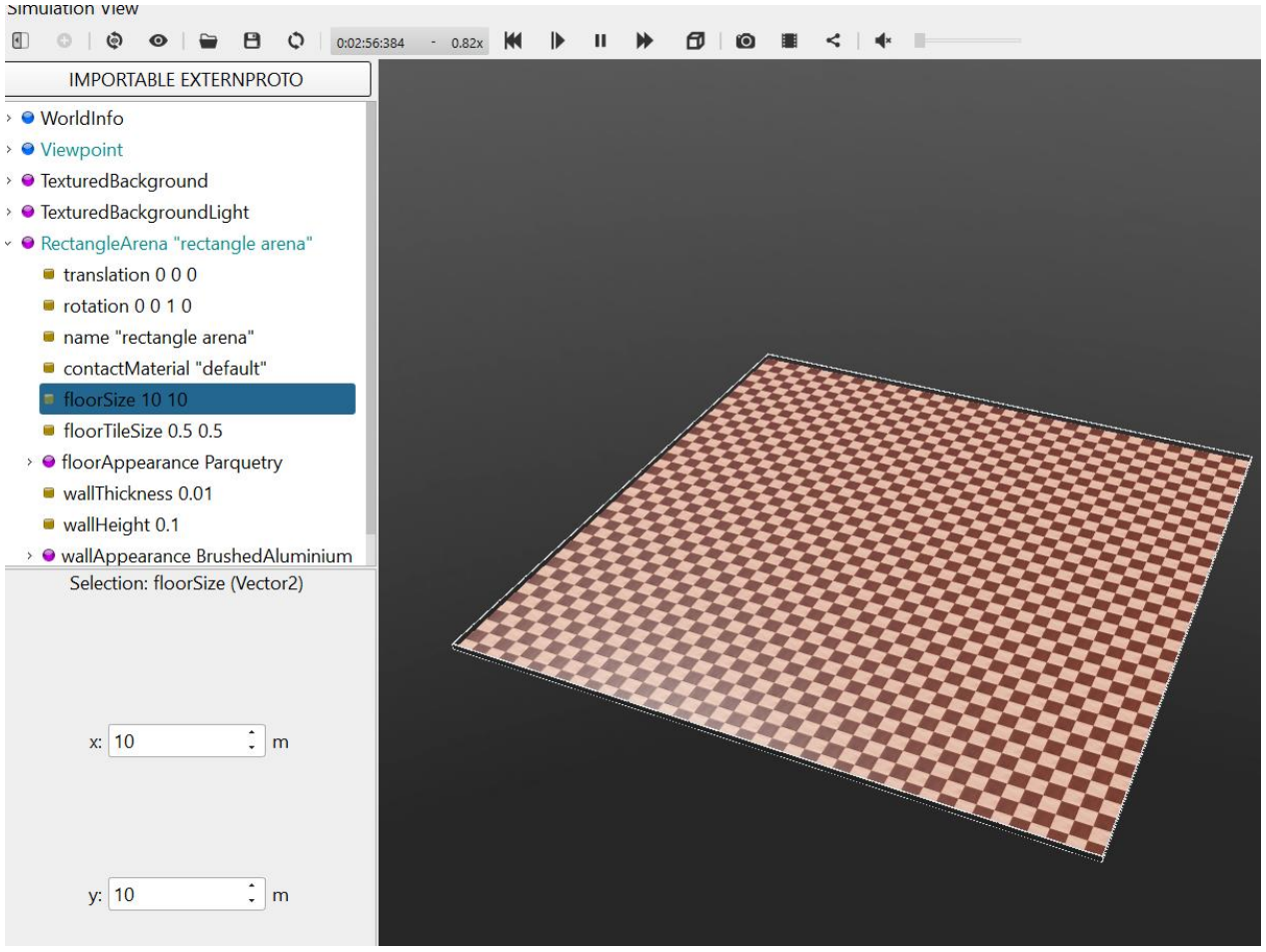
Day

01

Change the Floor Size

MIDOCEAN UNIVERSITY

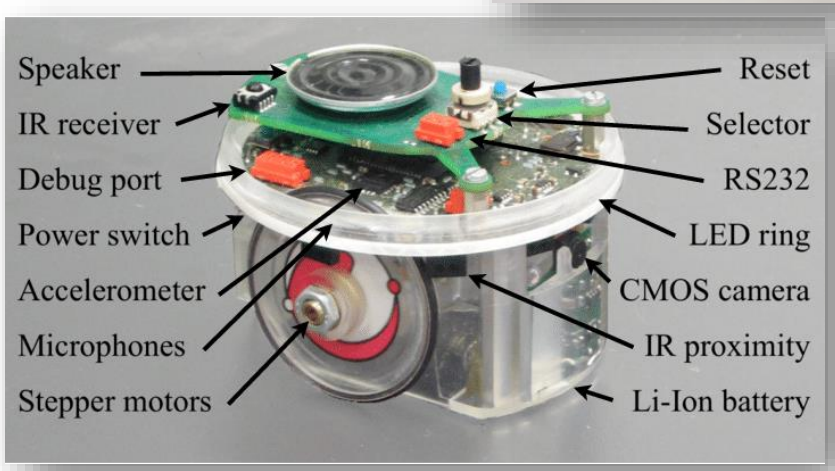




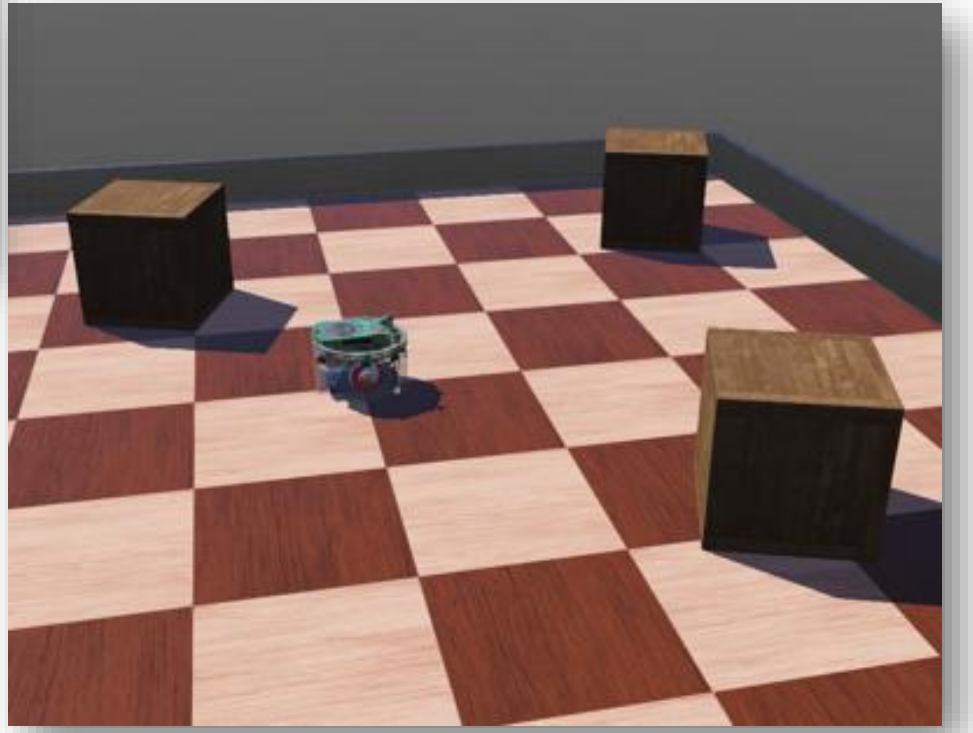
Day
01

NOW using Existing Objects and ROBOTS

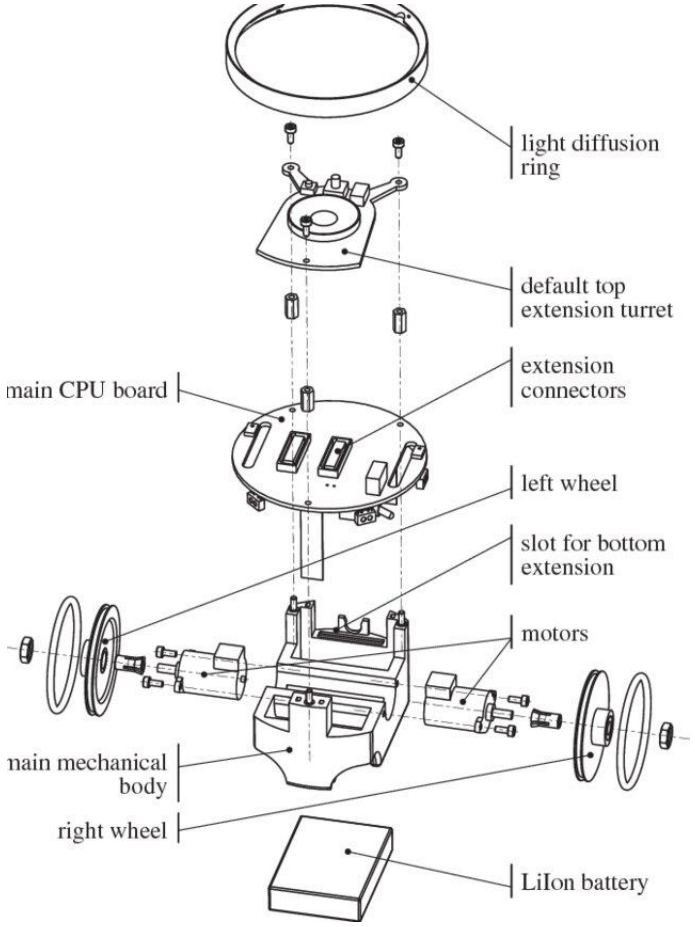
MIDOCEAN UNIVERSITY

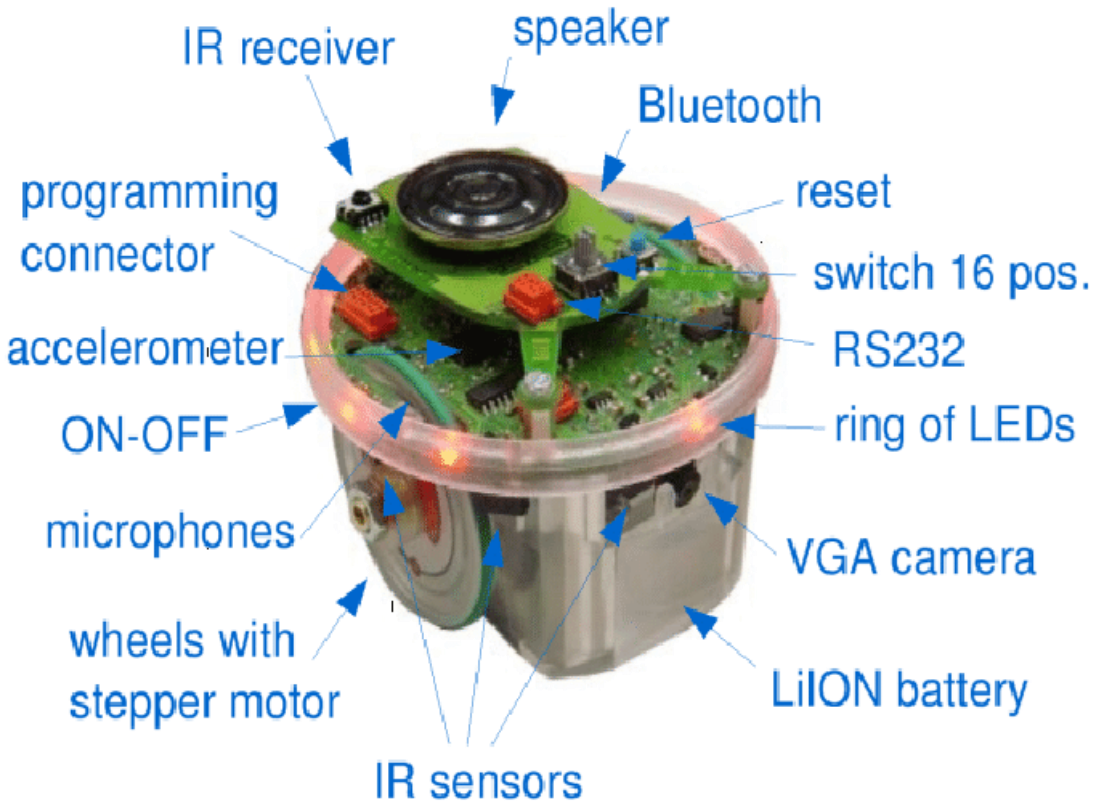


[e-puck education robot]

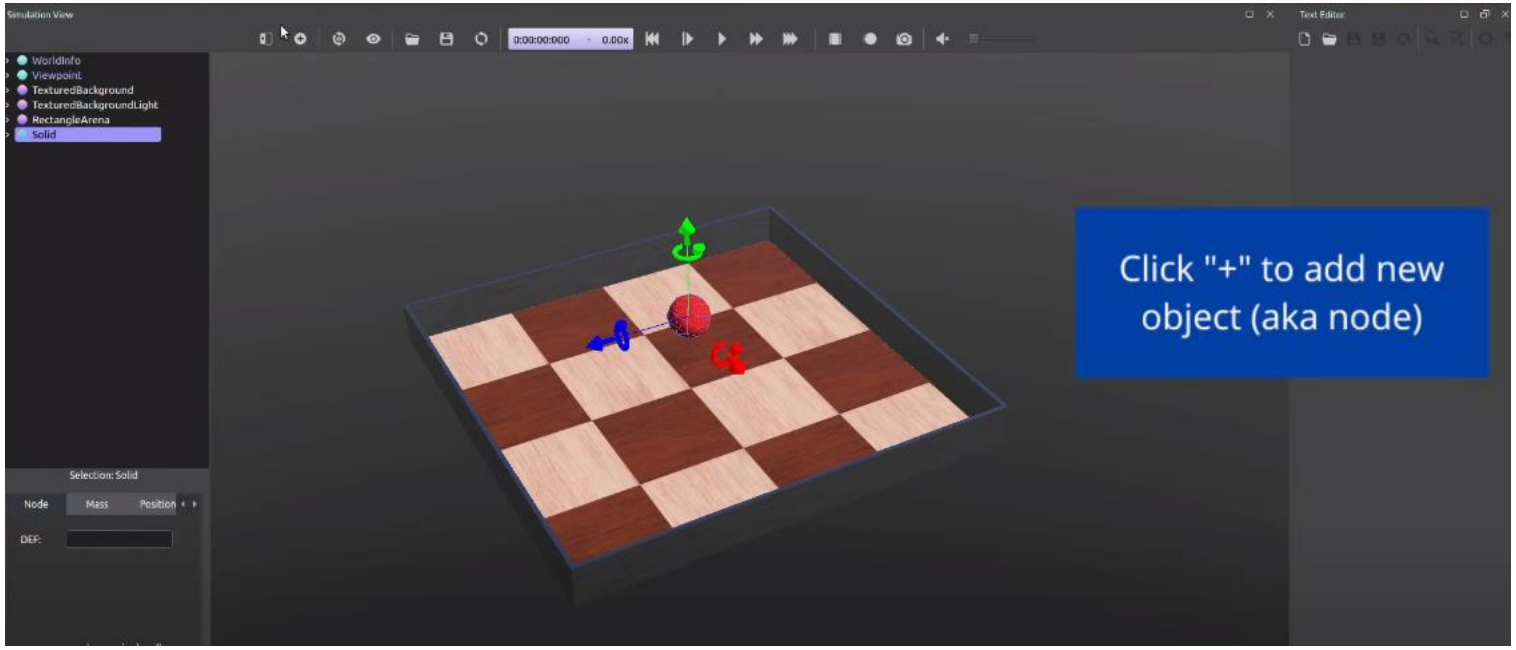


e-puck education robot





MIDOCEAN UNIVERSITY



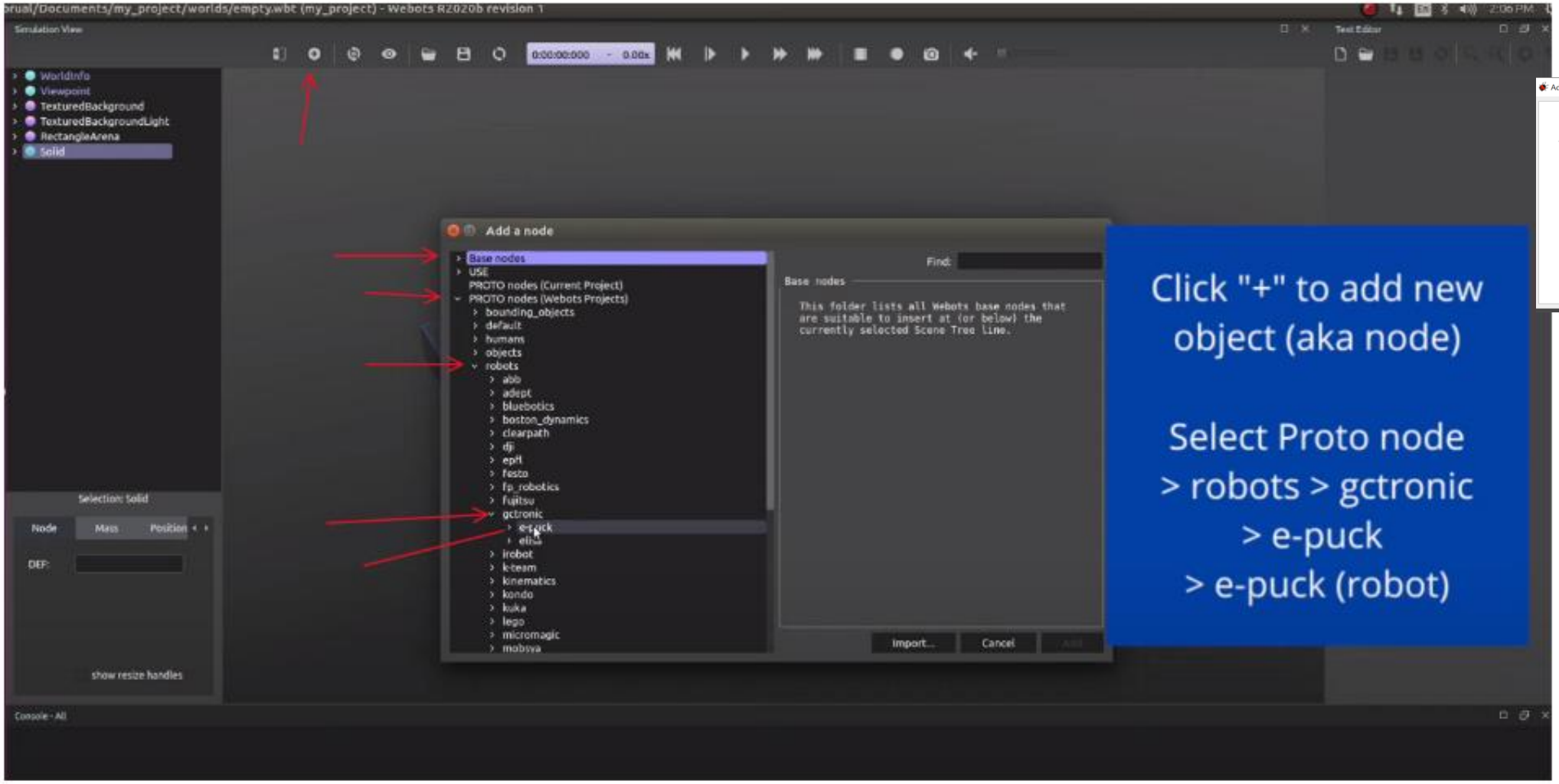
Click on the Add button at the top of the scene tree view. In the dialog box, choose PROTO nodes (Webots Projects) / robots / gctronic / e-puck / E-puck (Robot)

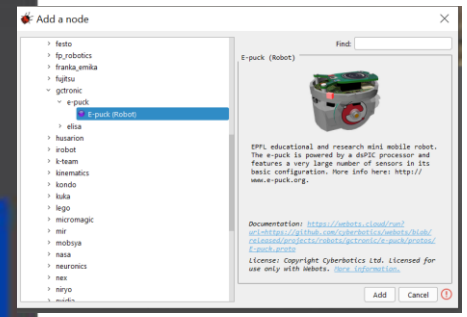
Day

01

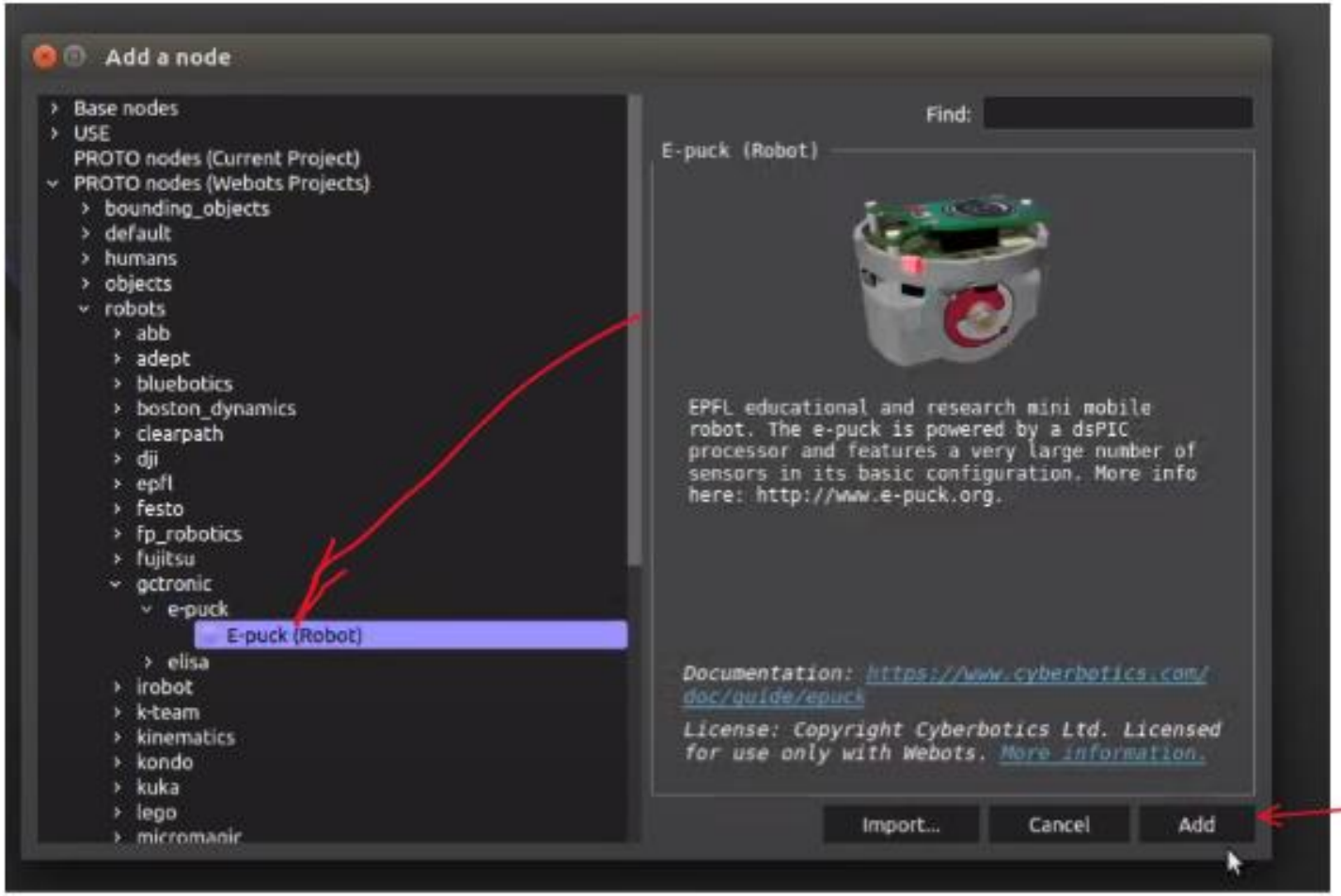
Follow the following selection sequences:

MIDOCEAN UNIVERSITY





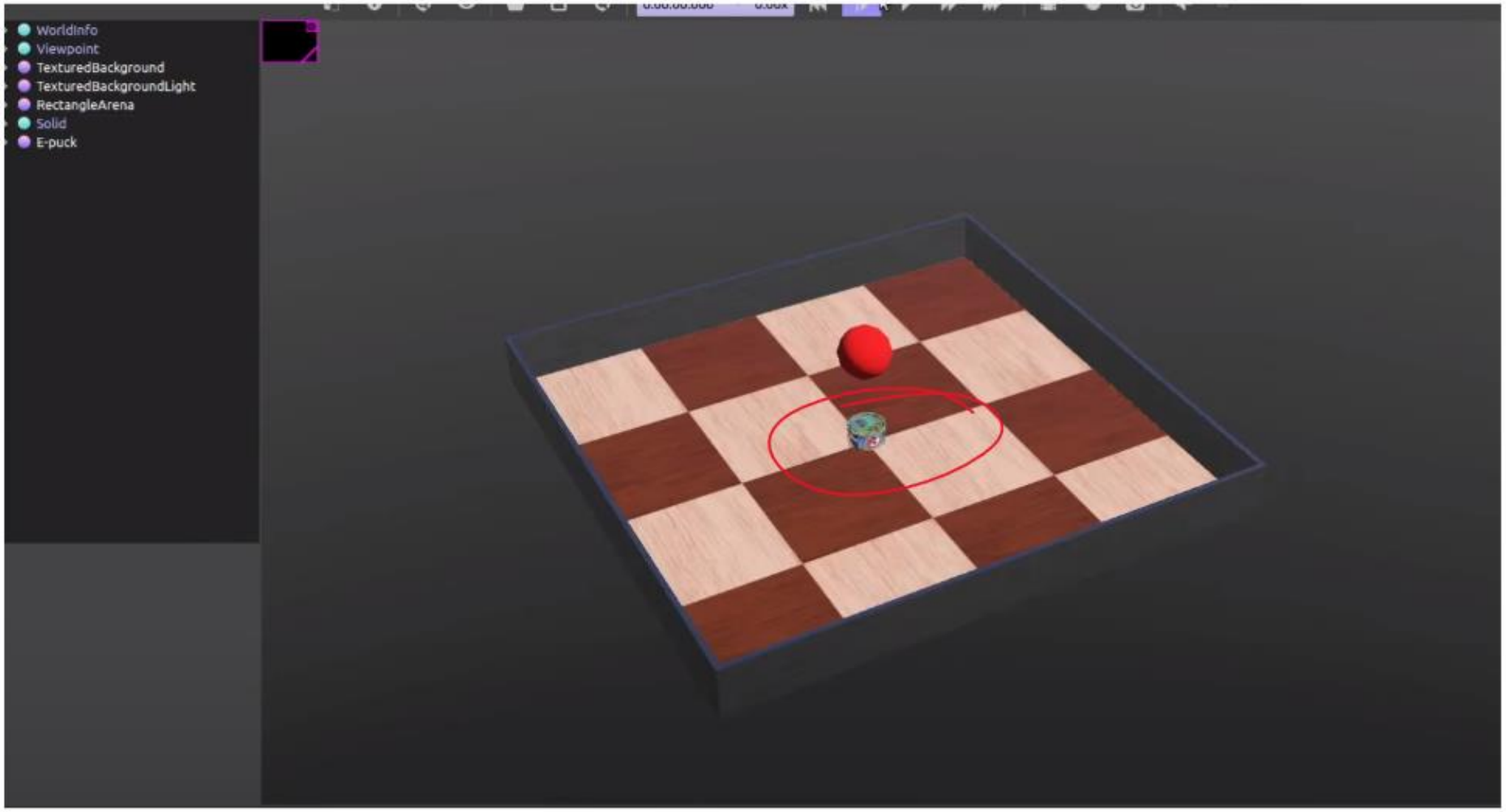
MIDOCEAN UNIVERSITY



Day

01

MIDOCEAN UNIVERSITY

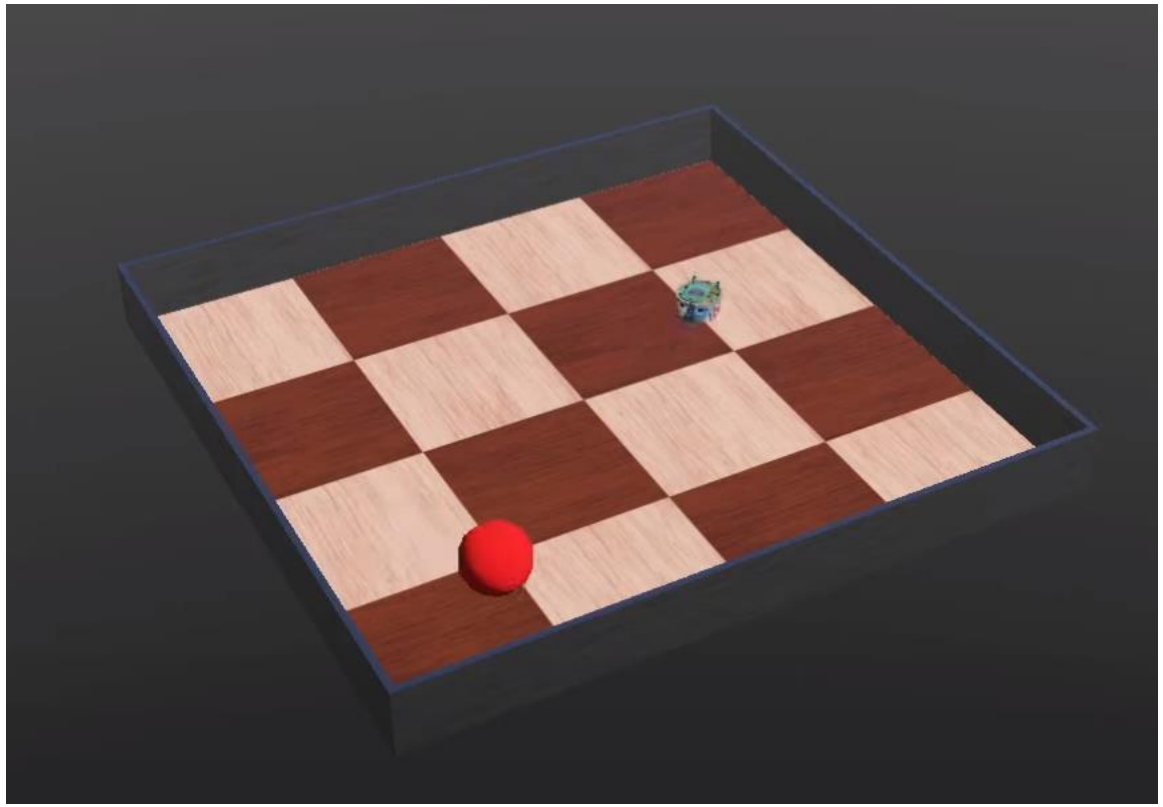


Day

01

When start the simulation (RUN) the Robot continue to drive.

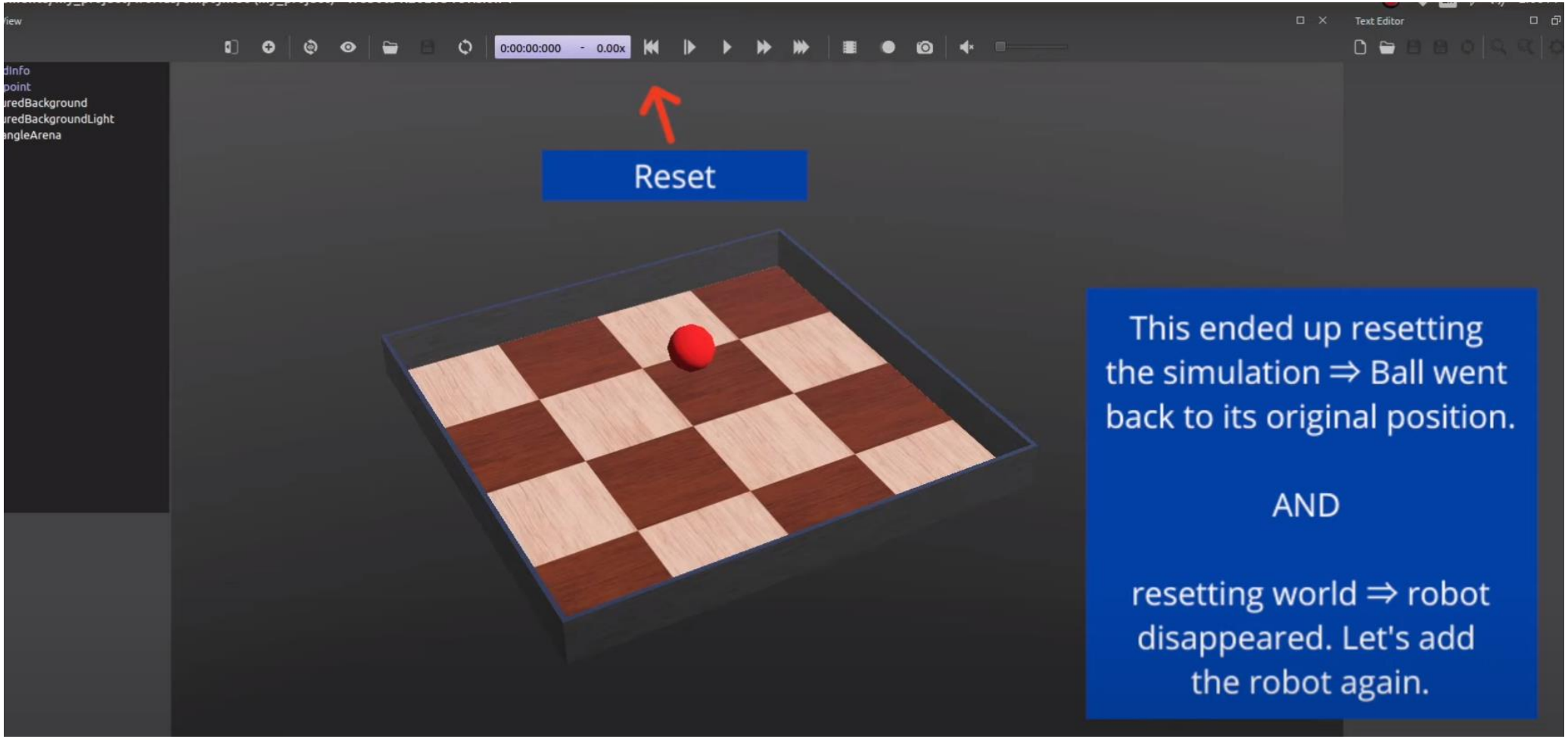
MIDOCEAN UNIVERSITY



Day

01

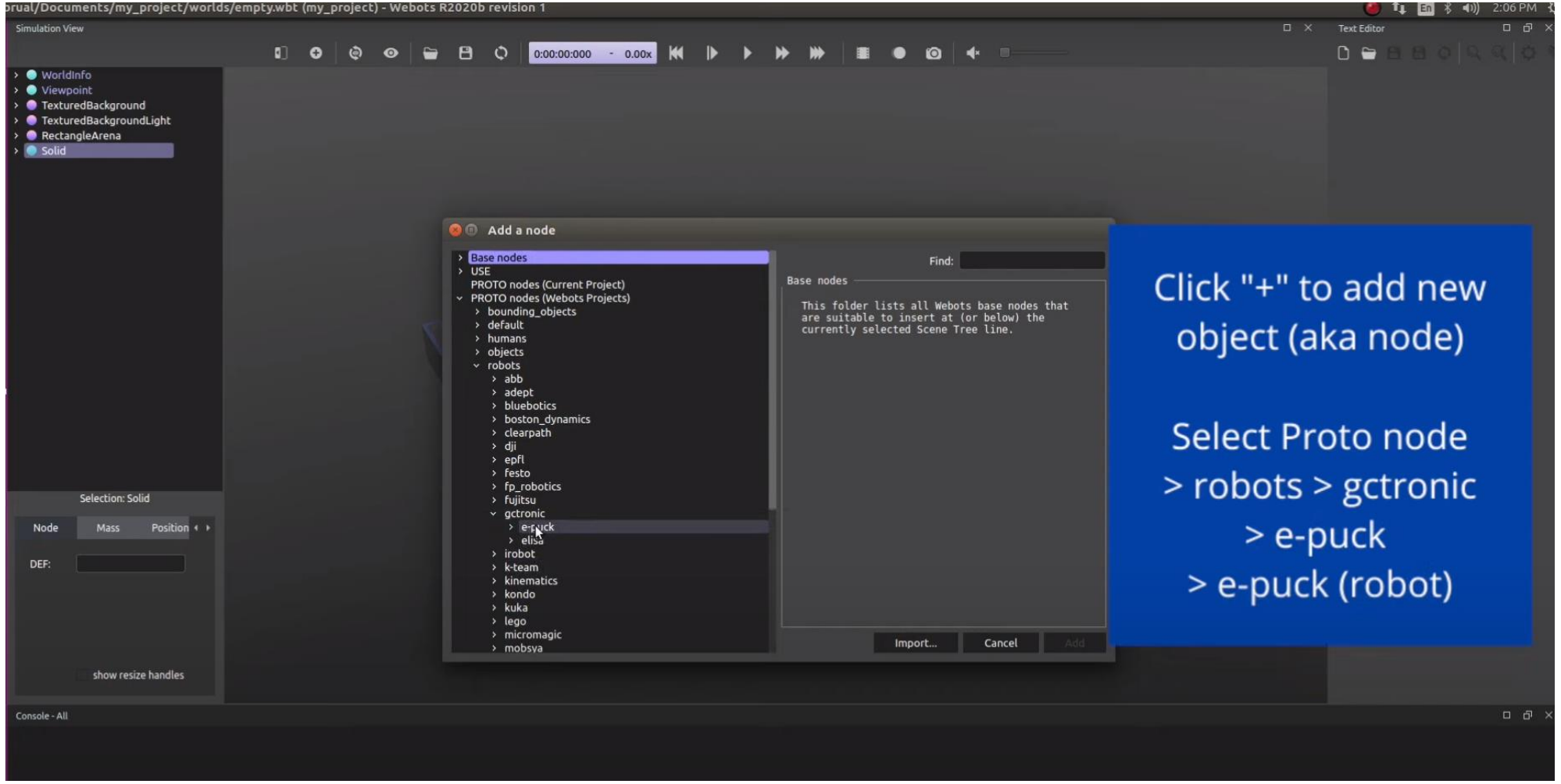
MIDOCEAN UNIVERSITY



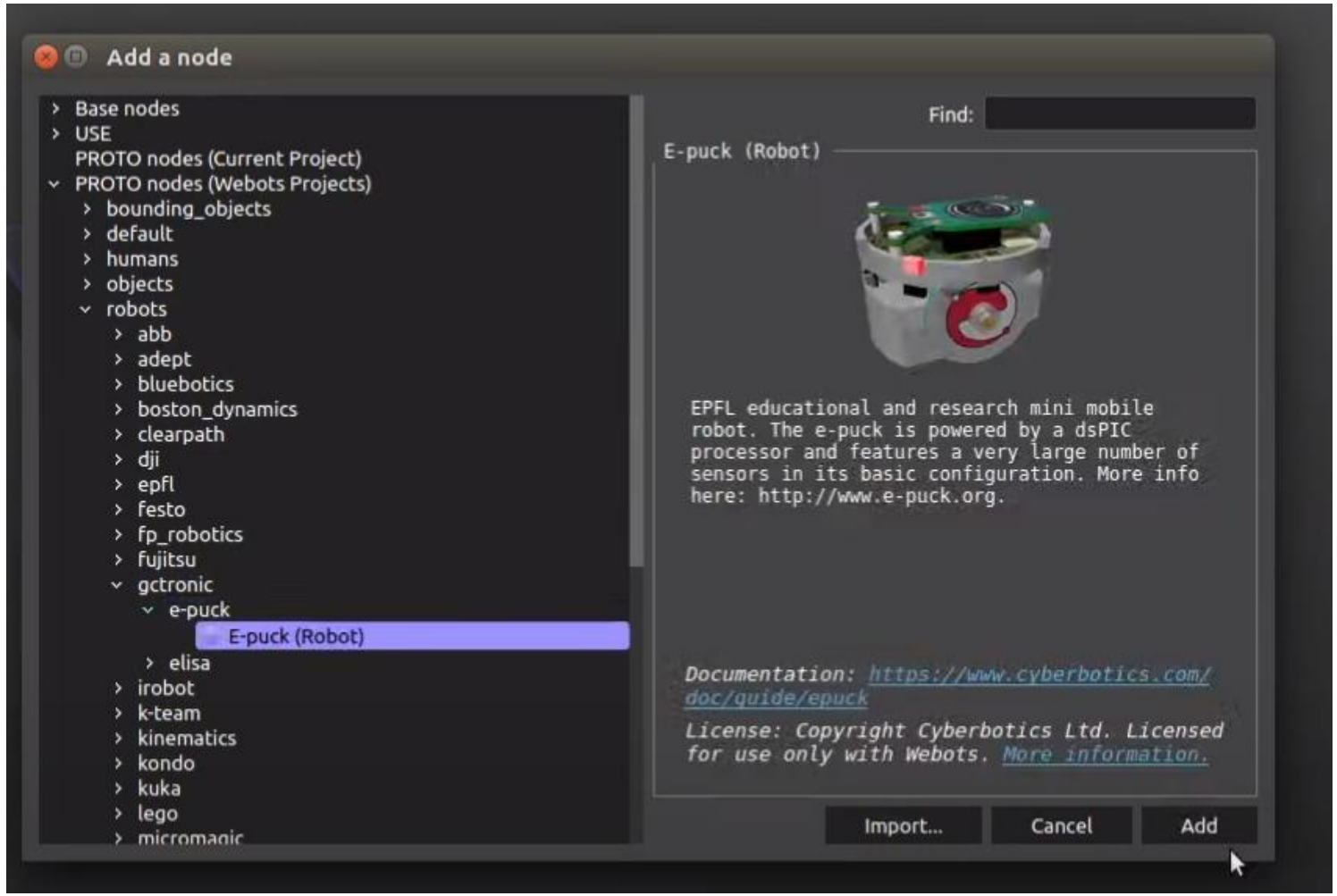
Day

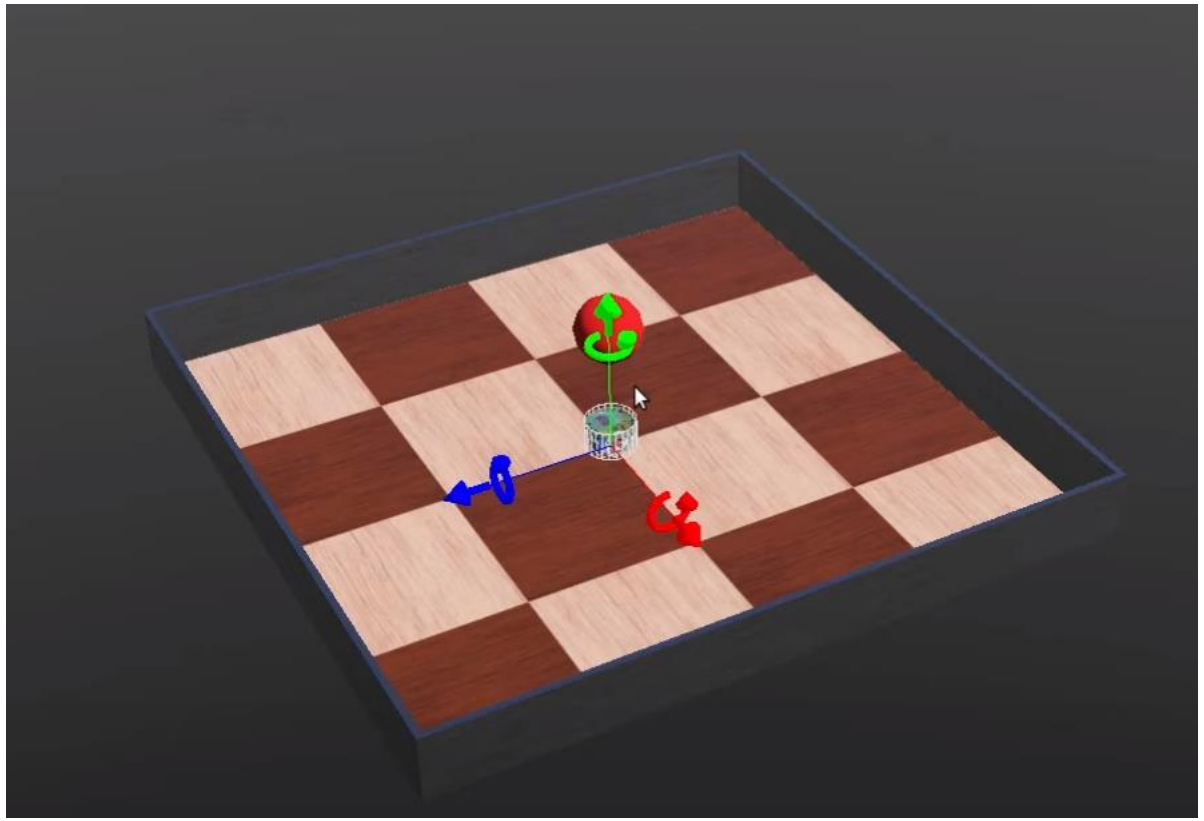
01

MIDOCEAN UNIVERSITY



MIDOCEAN UNIVERSITY



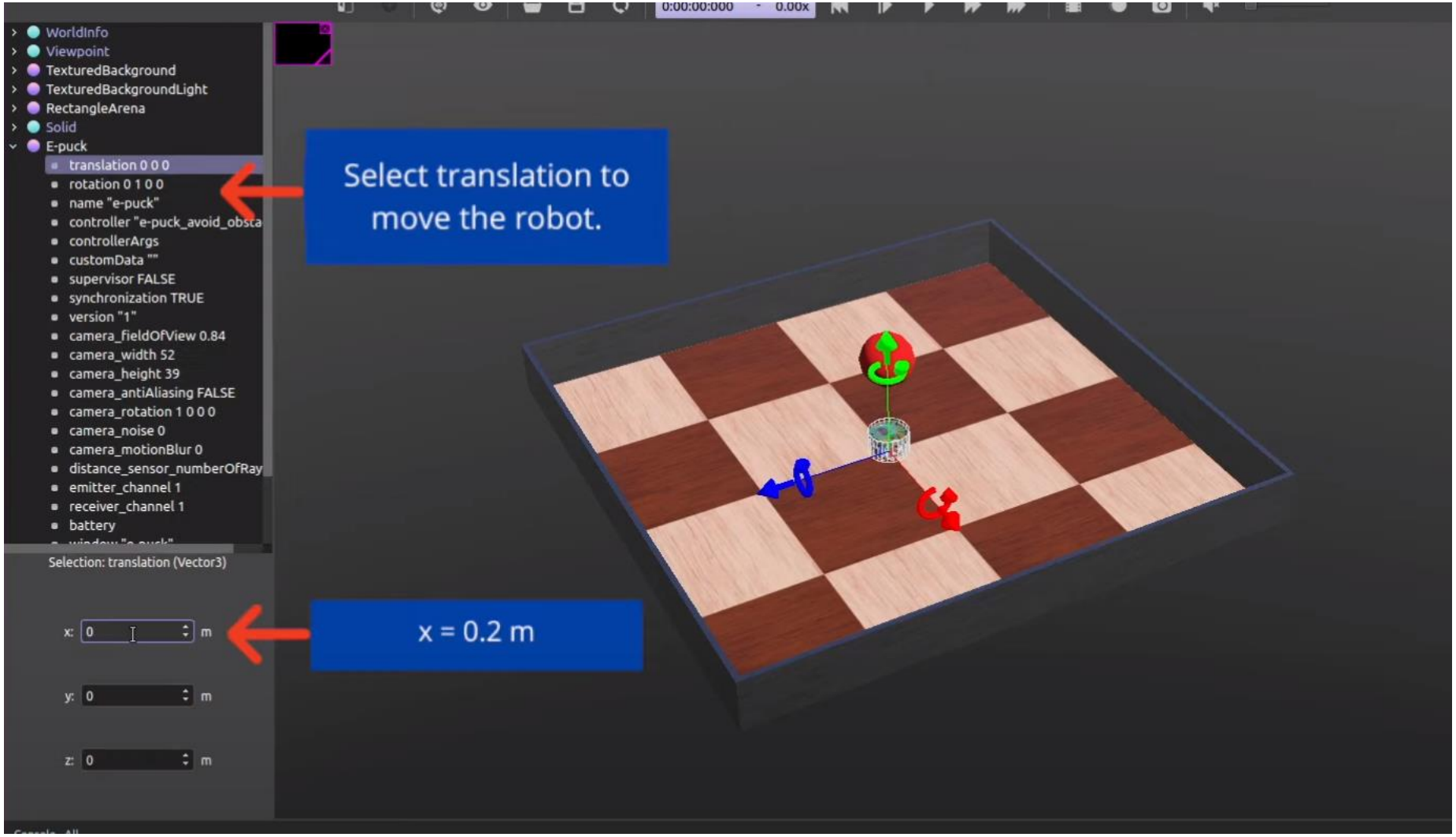


Day

01

Lets shift robot away from the original point (0,0,0):

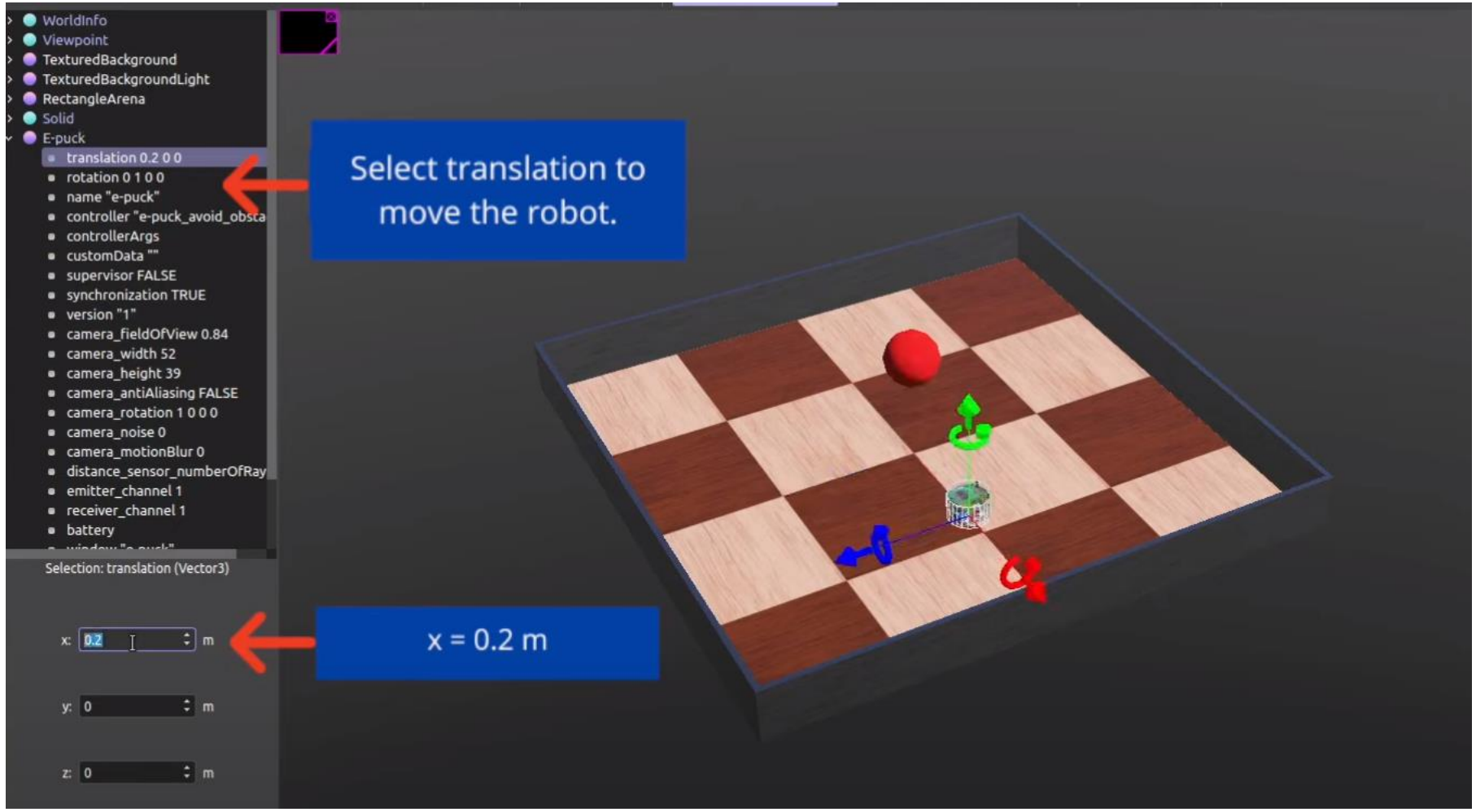
MIDOCEAN UNIVERSITY



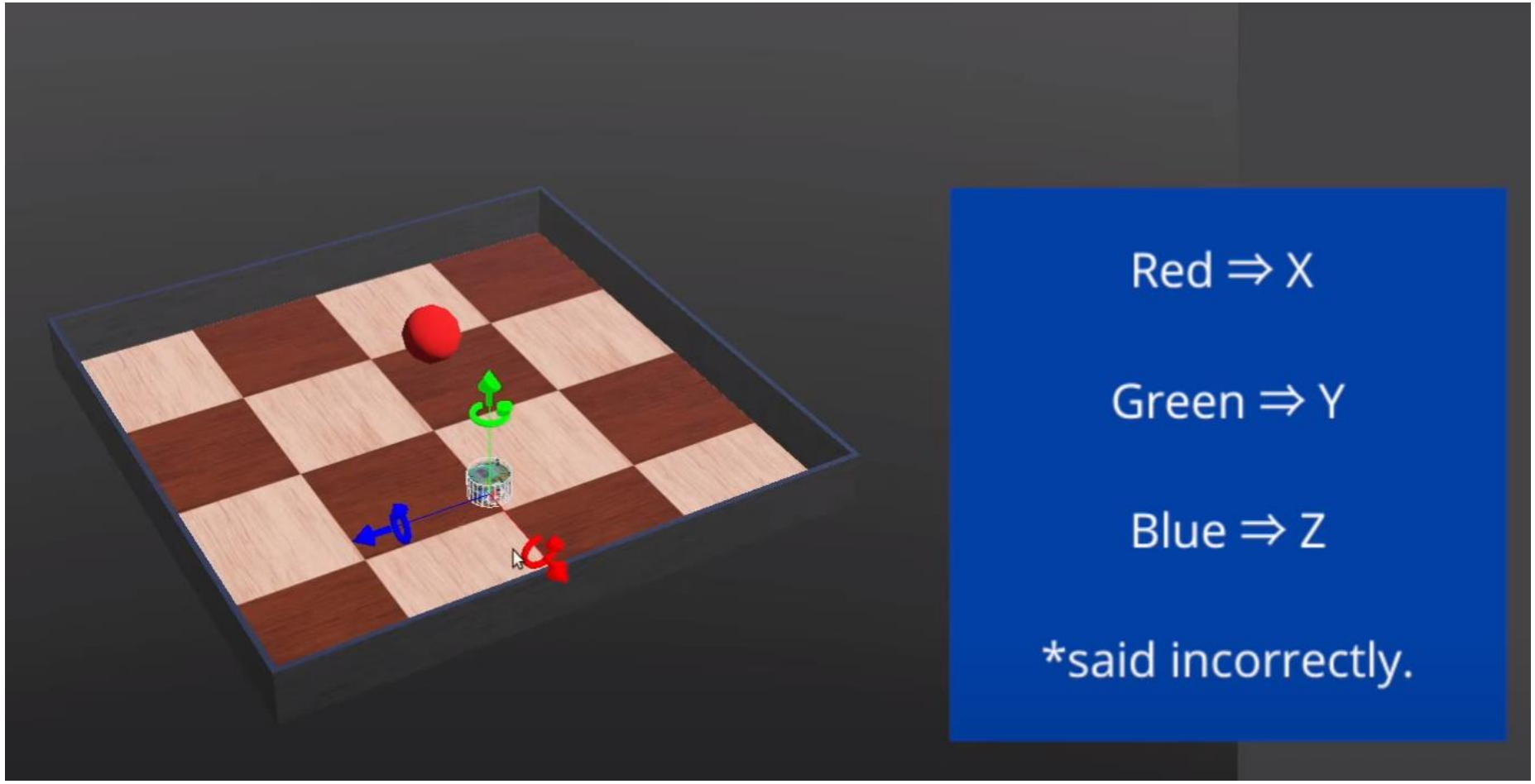
Day

01

MIDOCEAN UNIVERSITY



MIDOCEAN UNIVERSITY

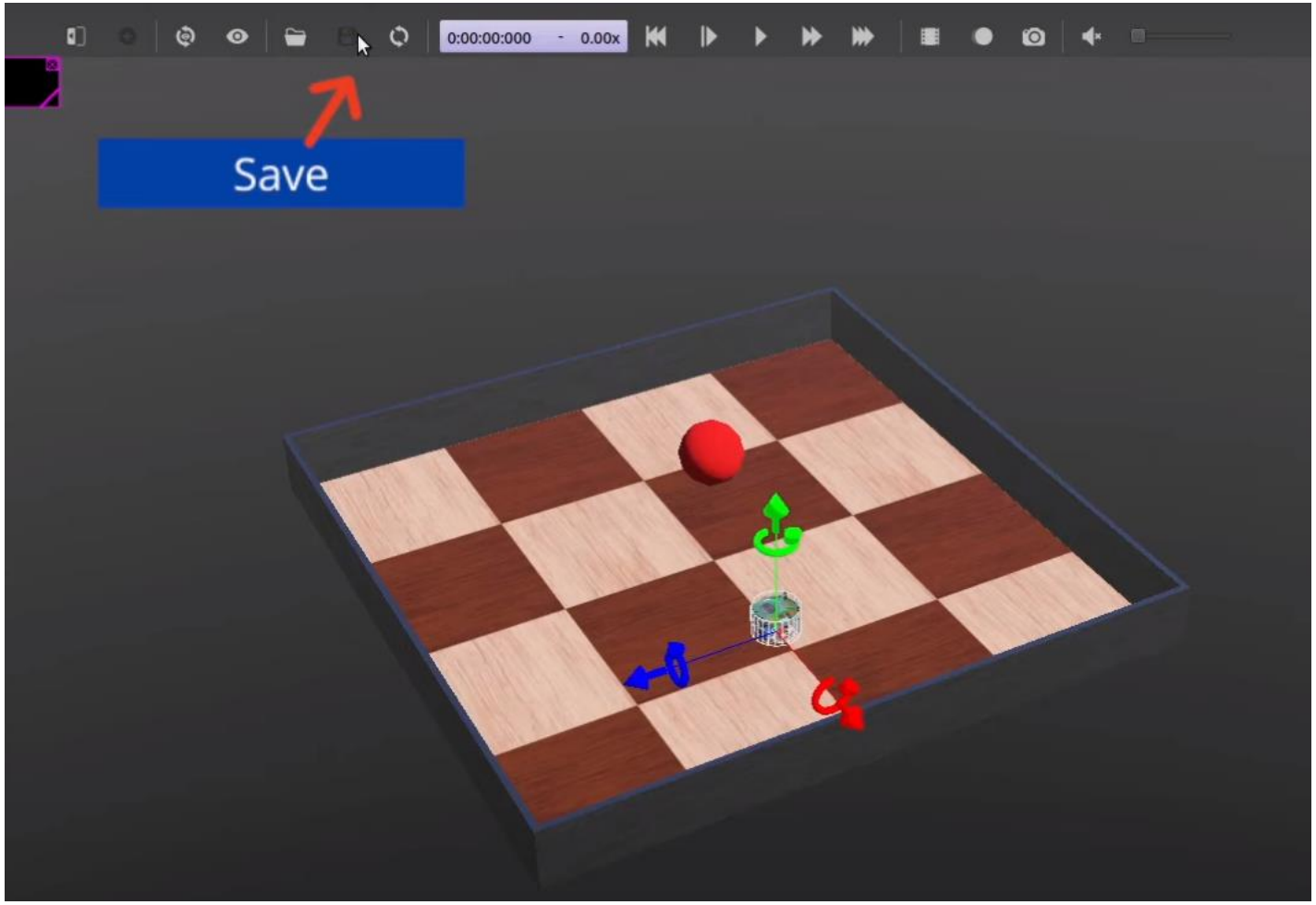


- The X-axis (Red Arrow) points forward from the robot.
- The Y-axis (Green Arrow) points to the left side of the robot.
- The Z-axis (Blue Arrow) points upwards from the robot.

Day

01

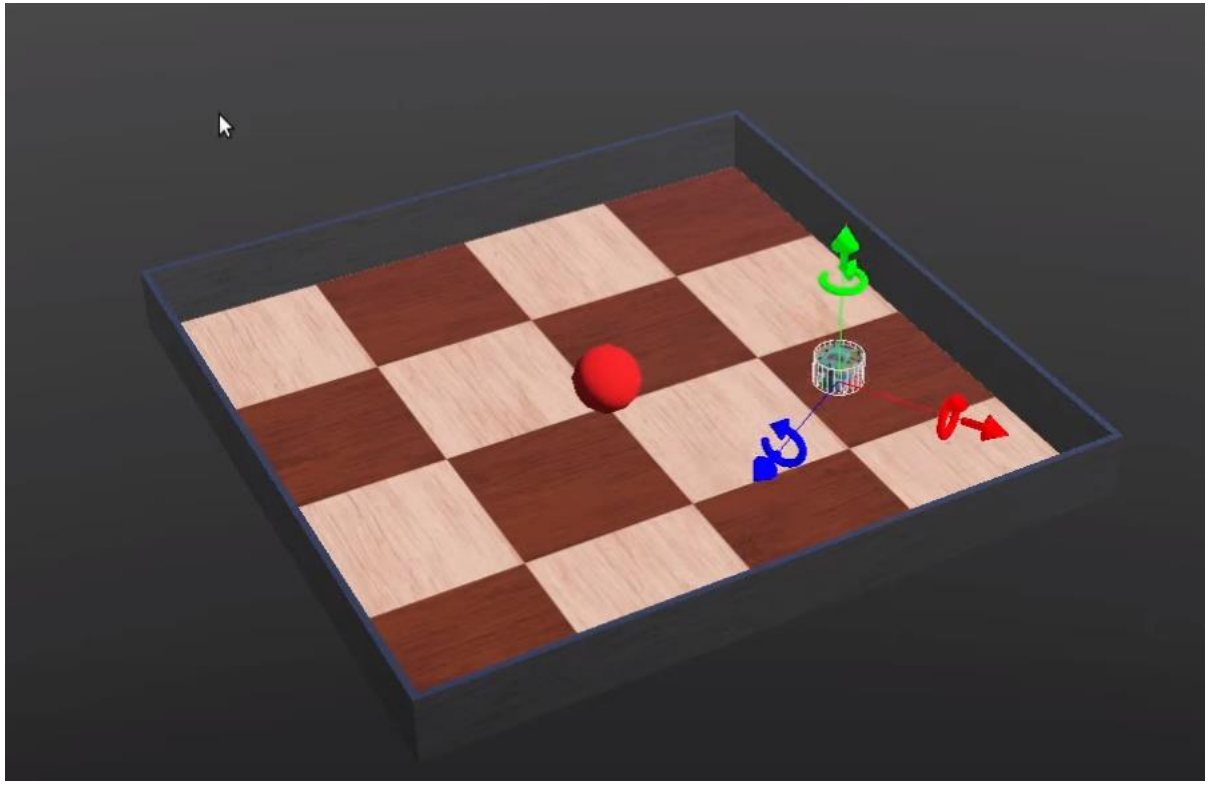
MIDOCEAN UNIVERSITY



RUN the Simulation now: you will see the robot move ?

Why?

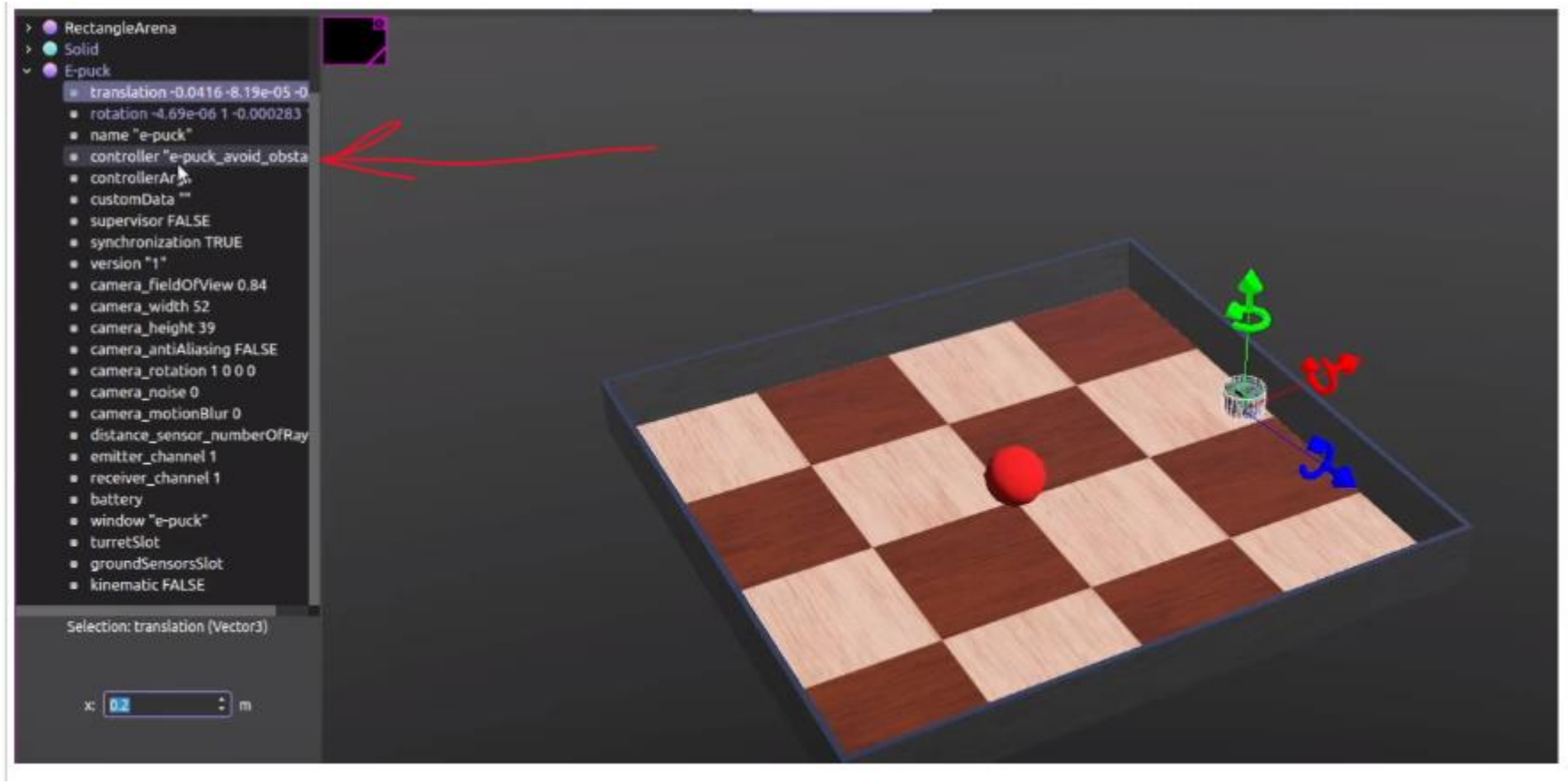
Because the robot comes with its controller.



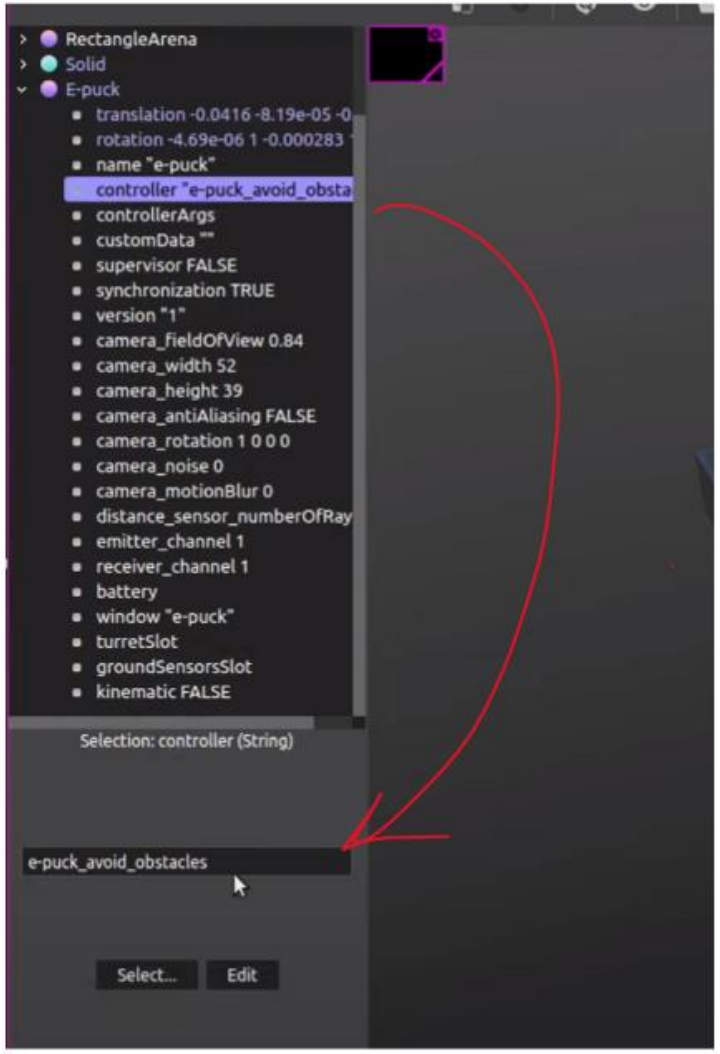
Day

01

Lets check the robot Controller:



Double click on the Controller:



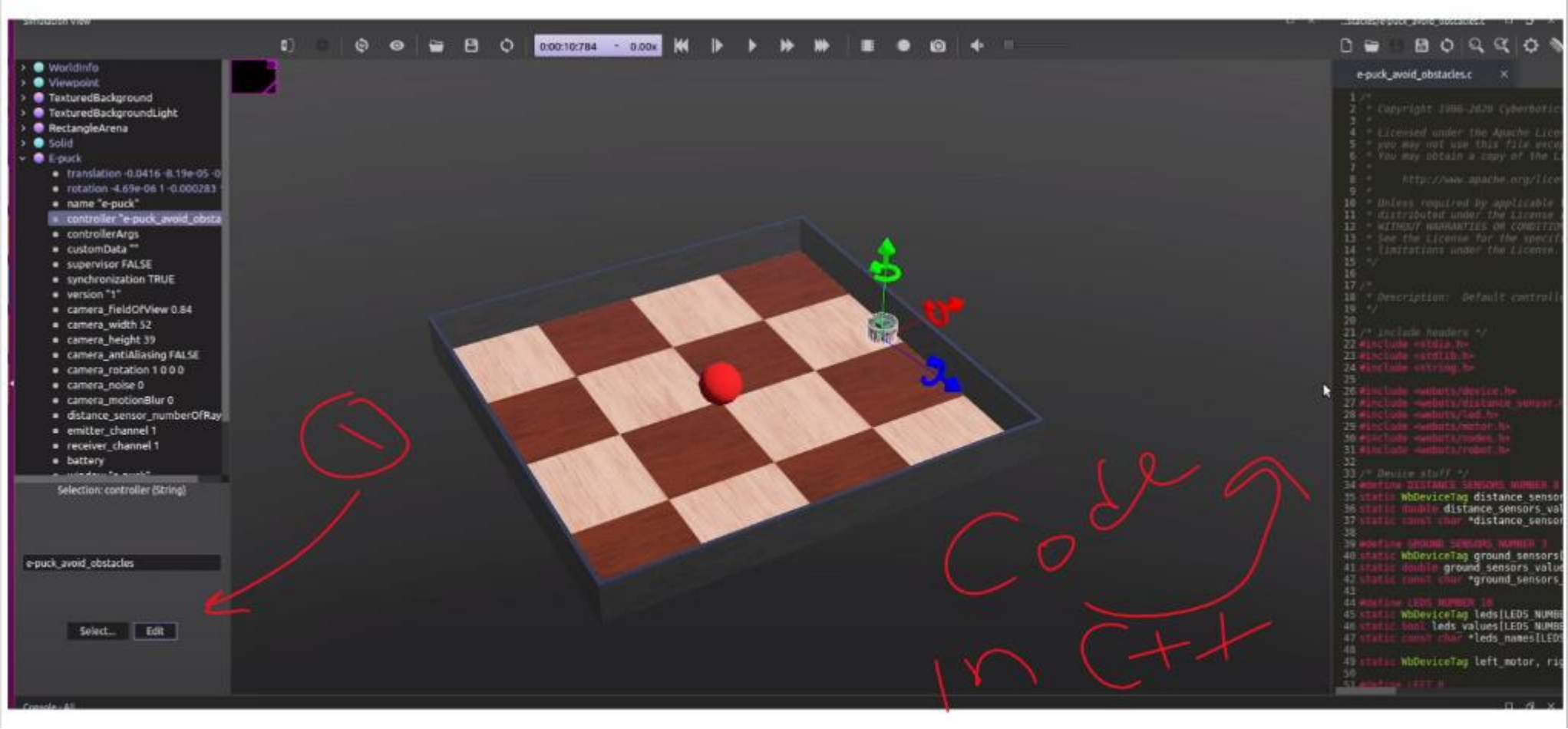
Day

01

NOW you can add your own code , Click on the EDIT:

The code is written in C++ but you can write Python code.

MIDOCEAN UNIVERSITY

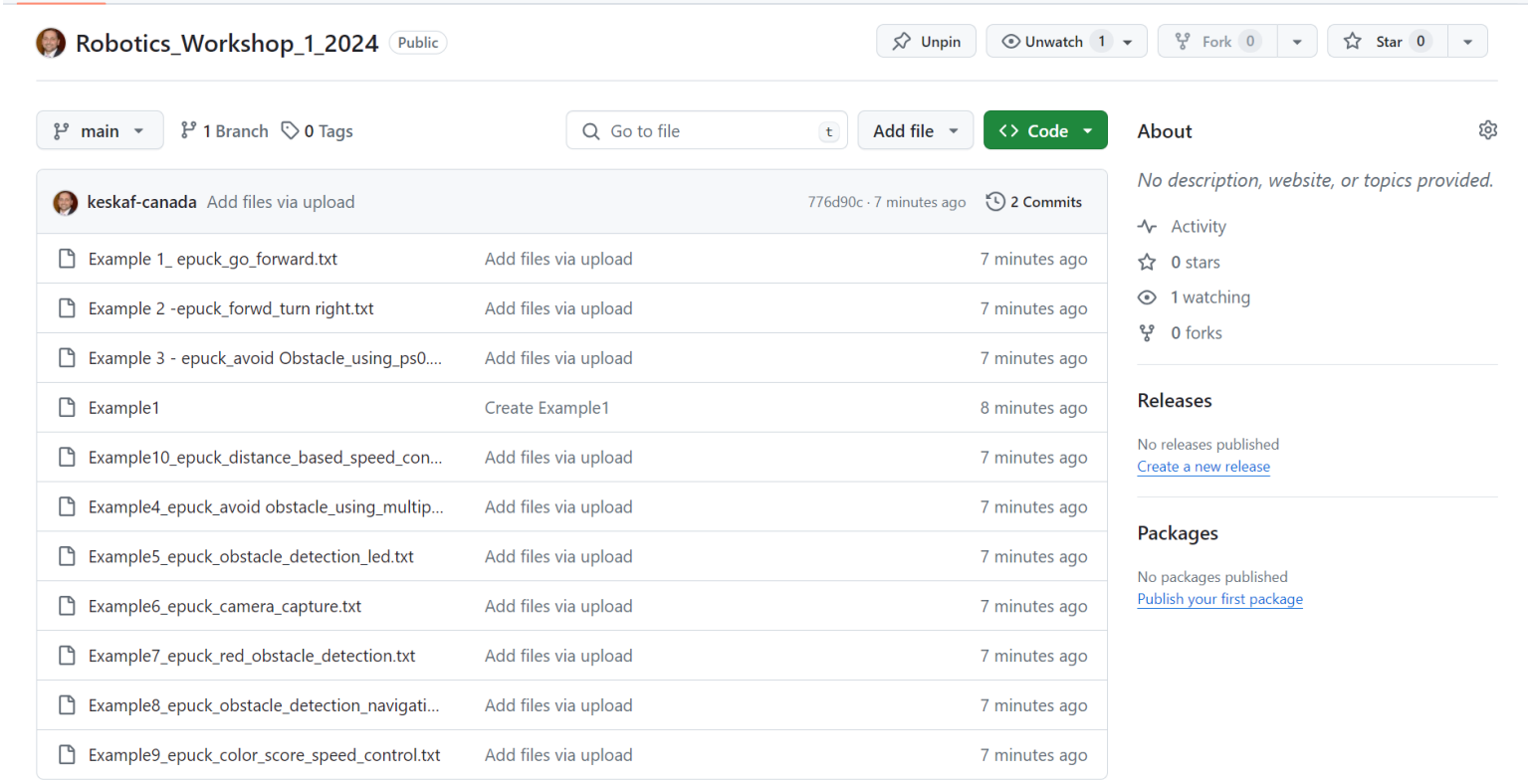


Day

01

MIDOCEAN UNIVERSITY

https://github.com/keskaf-canada/Robotics_Workshop_1_2024/tree/main



Create a New Controller

We will now program a simple controller that will just make the robot **move forwards**.
A **controller** is a program that defines the behavior of a robot.

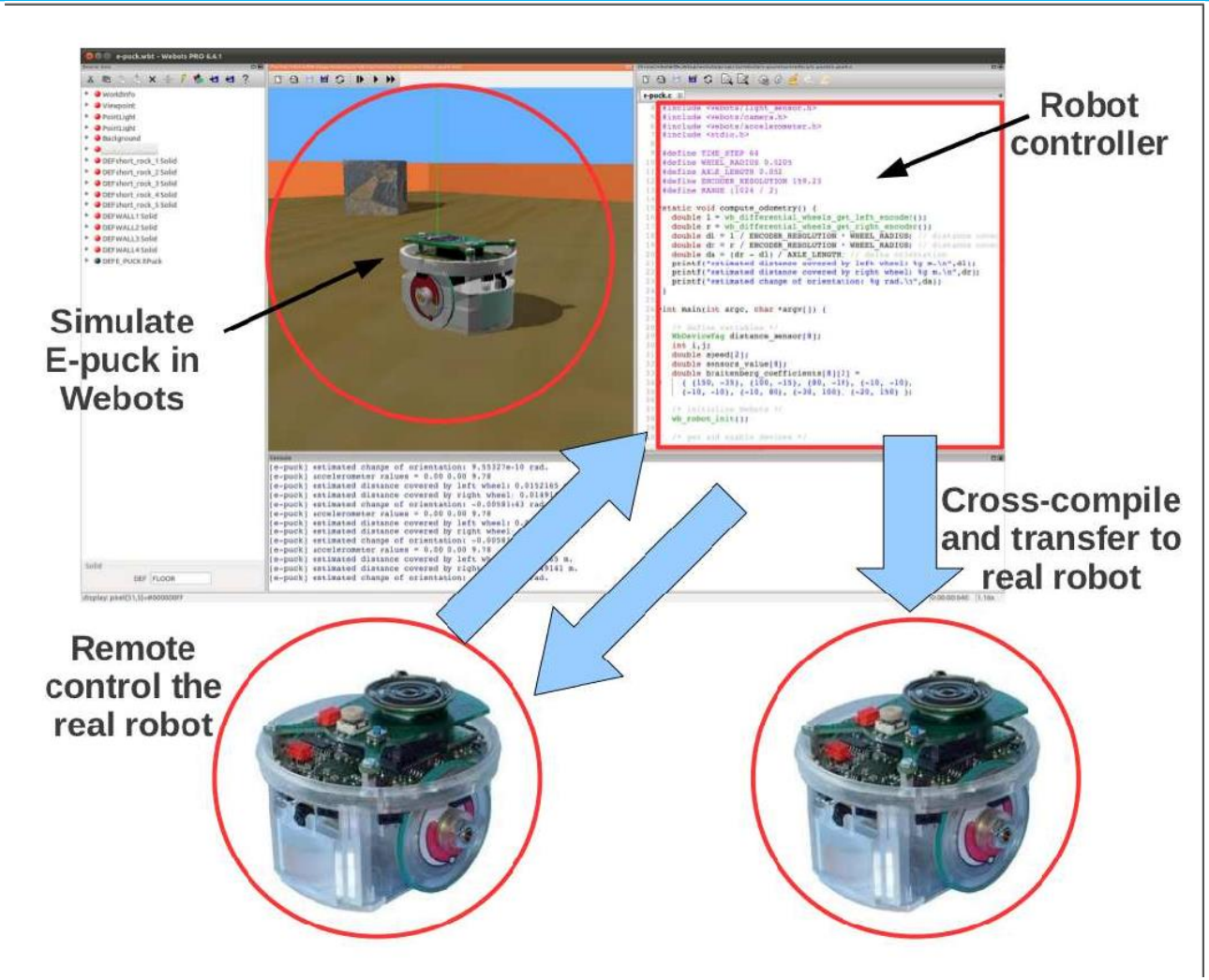
Webots controllers can be written in the following programming languages: C, C++, Java, Python, MATLAB, ROS, etc. C, C++ and Java controllers need to be compiled before they can be run as robot controllers.

Python and MATLAB controllers are interpreted languages so they will run without being compiled.

The controller field of a Robot node specifies which controller is currently associated to the robot. Note that the same controller can be used by several robots, but a robot can only use one controller at a time.

Each controller is executed in a separate child process usually spawned by Webots. Because they are independent processes, controllers don't share the same address space, and may run on different processor cores.

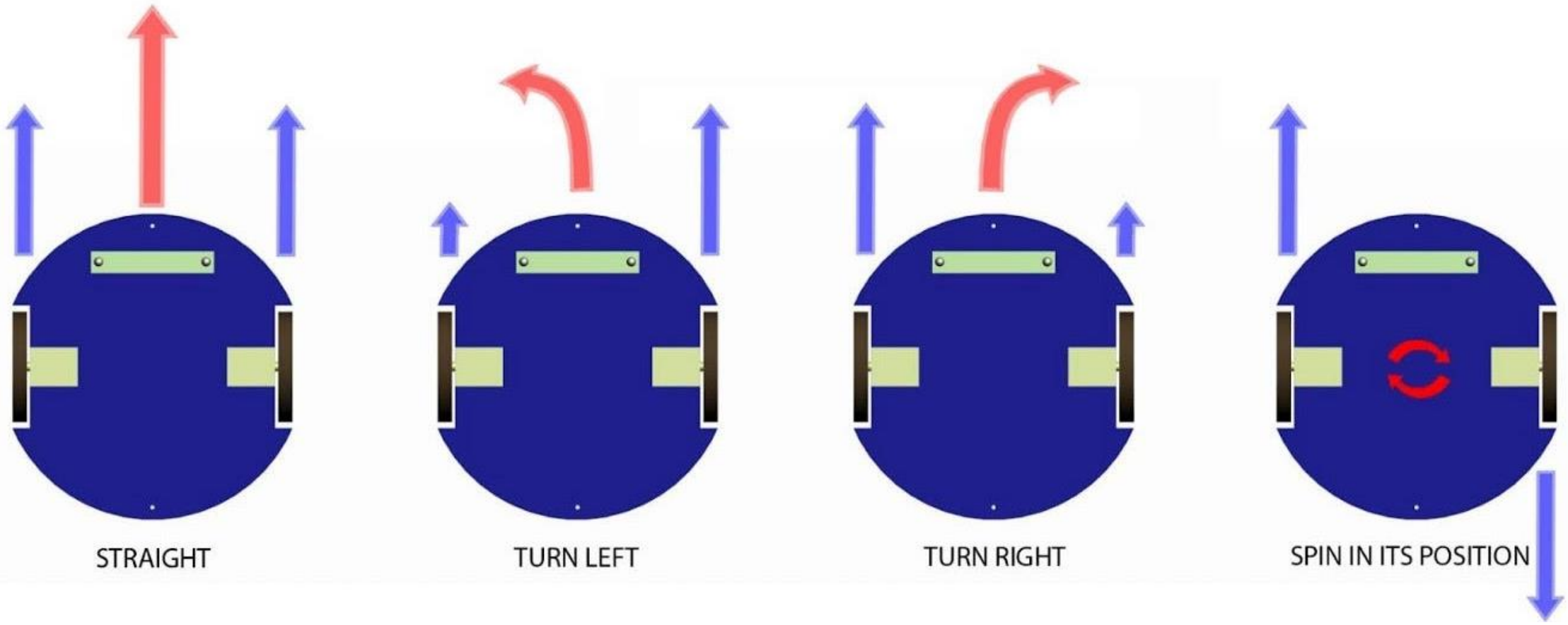
Create a New Controller



Hands-on #8: Create a new Python controller called epuck_go_forward using the **File / New / New Robot Controller...** menu item. This will create a new epuck_go_forward (or EPuckGoForward) directory in my_first_simulation/controllers. Select the option offering you to open the source file in the text editor.

Move forward

MIDOCEAN
UNIVERSITY



Day

01

Step-by-Step Guide

MIDOCEAN UNIVERSITY

Step 1: Set Up Your Webots Environment

1. **Open Webots:** Start Webots on your computer.
2. **Create a New Project:** If you don't already have a project, create a new one.
3. **Add the E-puck Robot:**
   - Go to the `Scene Tree` panel.
   - Right-click and select `Add...`
   - Navigate to `ROBOT` -> `E-puck` and add it to your scene.

Step 2: Associate a Controller with the Robot

1. **Select the E-puck Node:**
   - In the `Scene Tree` panel, find and click on the `E-puck` node.
2. **Assign a Controller:**
   - In the `Field` panel, find the `controller` field.
   - Click on the `Select...` button.
   - Choose `New File...`, name your controller file (e.g., `epuck_go_forward.py`), and confirm.

Step 3: Write the Controller Script

1. **Open the Controller Script:**
   - In your project directory, find and open the newly created Python controller script (e.g., `epuck_go_forward.py`).

Step 1: Set Up Your Webots Environment

1. **Open Webots:** Start Webots on your computer.
   - **افتح Webots:** ابدأ Webots على جهاز الكمبيوتر الخاص بك.
2. **Create a New Project:** If you don't already have a project, create a new one.
   - **إنشاء مشروع جديد:** إذا لم يكن لديك مشروع بالفعل، قم بإنشاء مشروع جديد.
3. **Add the E-puck Robot:**
   - **إضافة الروبوت E-puck:**
   - Go to the `Scene Tree` panel.
     - انتقل إلى لوحة `Scene Tree`.
   - Right-click and select `Add...`
     - انقر بزر الماوس الأيمن واختر `Add...`
   - Navigate to `ROBOT` -> `E-puck` and add it to your scene.
     - انتقل إلى `ROBOT` -> `E-puck` وأضفه إلى المشهد الخاص بك.

Step 2: Associate a Controller with the Robot

1. **Select the E-puck Node:**
   - **حدد العقدة E-puck:**
   - In the `Scene Tree` panel, find and click on the `E-puck` node.
     - في لوحة `Scene Tree`، ابحث عن العقدة `E-puck` وانقر عليها.
2. **Assign a Controller:**
   - **تعيين وحدة تحكم:**
   - In the `Field` panel, find the `controller` field.
     - في لوحة `Field`، ابحث عن حقل `controller`.
   - Click on the `Select...` button.
     - انقر على الزر `Select...`.
   - Choose `New File...`, name your controller file (e.g., `epuck_go_forward.py`), and confirm.
     - اختر `New File...`، قم بتسمية ملف وحدة التحكم الخاص بك (مثلاً، `epuck_go_forward.py`) وقم بالتأكيد.

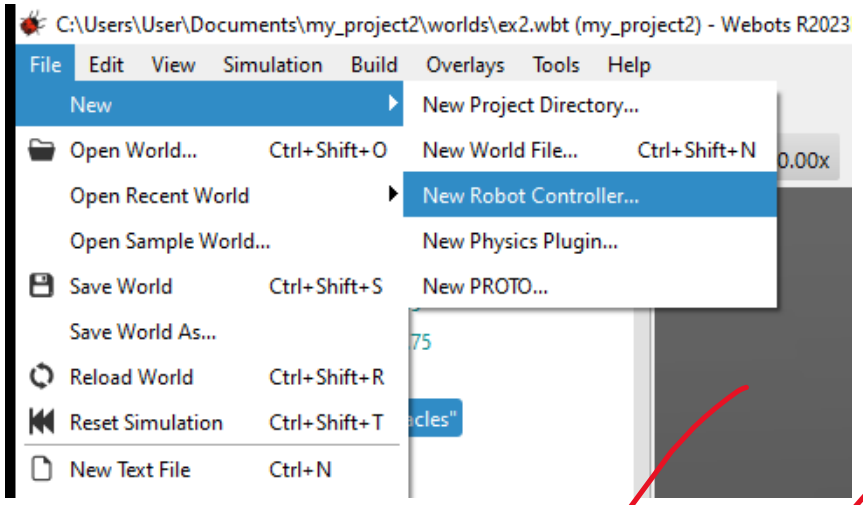
Step 3: Write the Controller Script

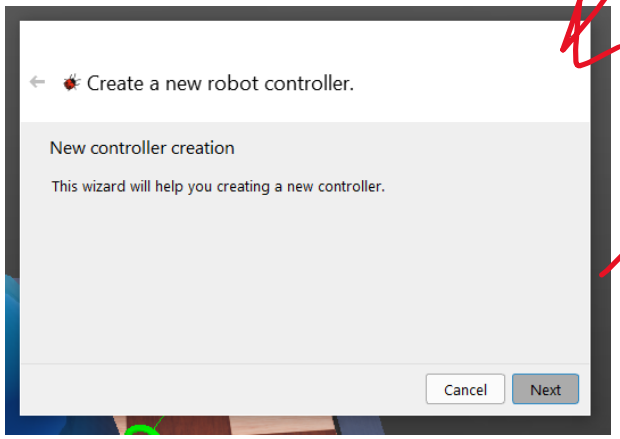
1. **Open the Controller Script:**
    - **افتح برنامج وحدة التحكم:**
    - In your project directory, find and open the newly created Python controller script (e.g., `epuck_go_forward.py`).
        - في دليل مشروعك، ابحث وافتح ملف وحدة التحكم Python الذي تم إنشاؤه حديثاً (مثلاً، `epuck_go_forward.py`).
2. **Write the Script:**
    - **اكتب البرنامج:**
    - Copy and paste the following simplified code into your controller script. Each step includes detailed comments to help you understand what each part does.
        - انسخ والصق الكود المبسط التالي في برنامج وحدة التحكم الخاص بك. يتضمن كل خطوة تعليقات مفصلة لمساعدتك على فهم كل جزء.

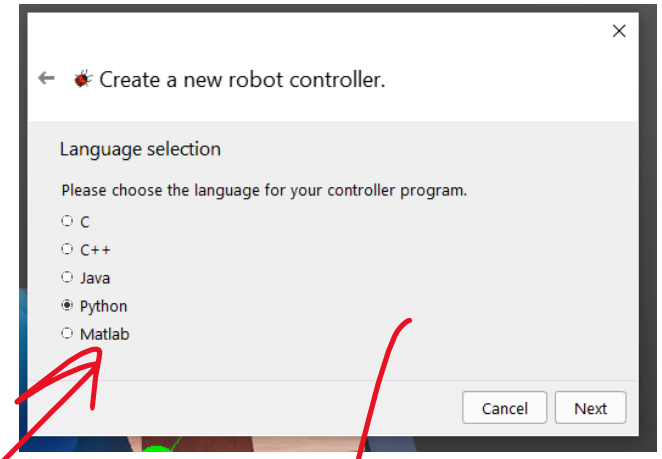
Day

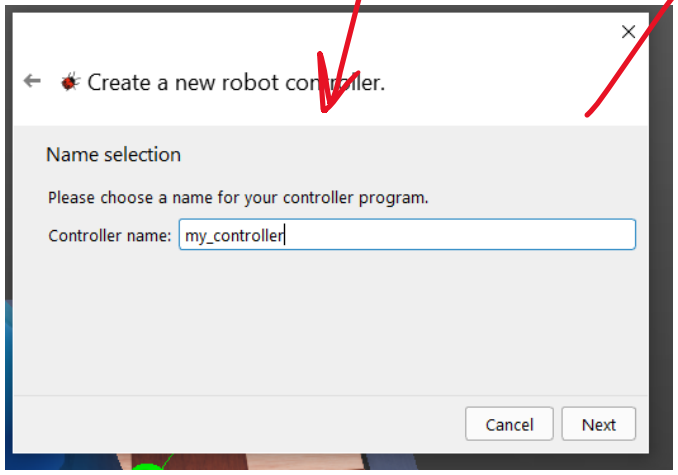
01

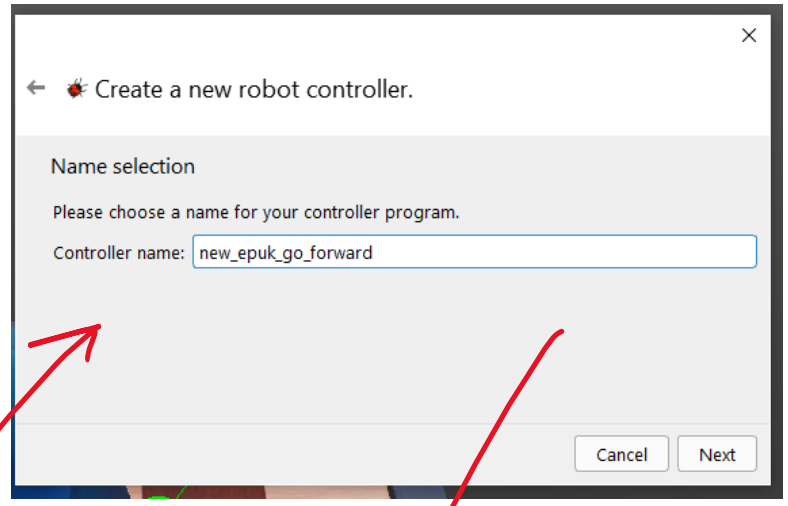
MIDOCEAN UNIVERSITY

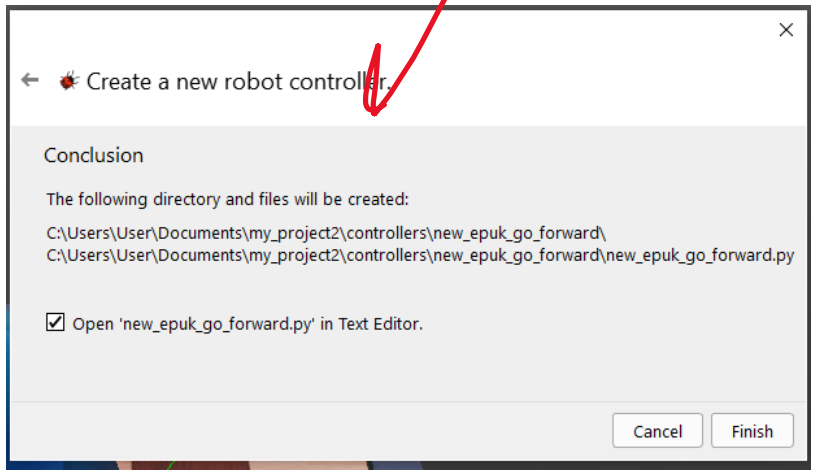








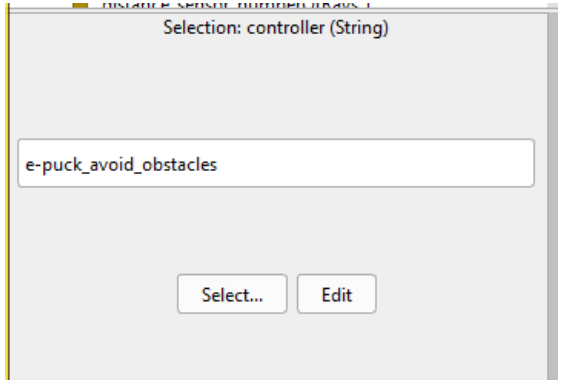


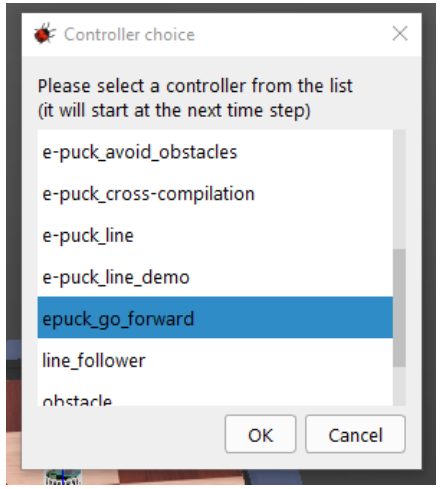


MIDOCEAN UNIVERSITY

The new source file is displayed in Webots text editor window. We will now associate new epuck_go_forward controller to the E-puck node.

In the scene tree view, select the controller field of the E-puck node, then use the field editor at the bottom of the Scene Tree view: press the Select... button and then select epuck_go_forward in the list. Once the controller is associated with the robot, save the world. Modify the program by getting the motor devices (leftMotor = robot.getDevice('left wheel motor')), and by applying a motor command (leftMotor.setPosition(10.0)):





Day
01

MIDOCEAN
UNIVERSITY

code

Day
01

MIDOCEAN UNIVERSITY

```
# Import the Robot and Motor classes from the Webots controller module
from controller import Robot, Motor
# استيراد الفئات Robot و Motor من وحدة التحكم Webots

# Define a time step for the simulation (this value should match Webots simulation time step)
TIME_STEP = 64
# تعريف خطوة زمنية للمحاكاة (يجب أن تطابق هذه القيمة خطوة الوقت في Webots)

# Define the maximum speed for the e-puck robot motors
MAX_SPEED = 6.28  # Note: e-puck's speed is often specified in radians per second
# تحديد السرعة القصوى لمحركات الروبوت e-puck
# ملاحظة: غالبًا ما يتم تحديد سرعة e-puck في الثانية بالراديان
```

MIDOCEAN UNIVERSITY

```
# Create an instance of the Robot class
robot = Robot()
#  إنشاء مثيل لفئةRobot

# Get the left and right wheel motors of the e-puck robot
leftMotor = robot.getDevice('left wheel motor')
rightMotor = robot.getDevice('right wheel motor')
#  الحصول على محركات العجلة اليسرى واليمنى للروبوتe-puck

# Set the target position of the motors to infinity (to control speed instead of position)
leftMotor.setPosition(float('inf'))
rightMotor.setPosition(float('inf'))
# (ضبط الموضع المستهدف للمحركات إلى ما لا نهاية (للتحكم في السرعة بدلاً من الموضع
```

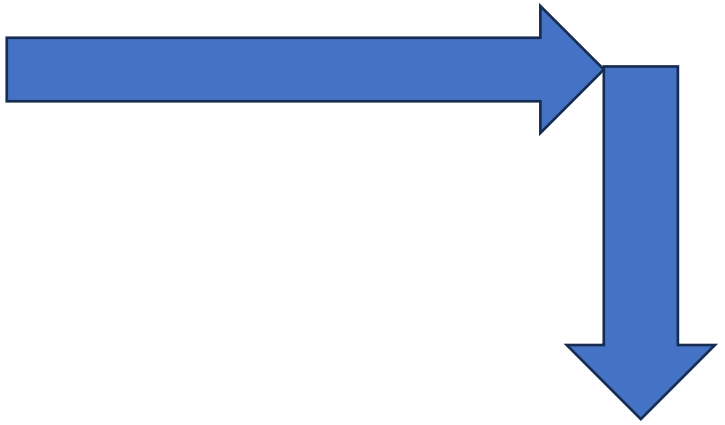
MIDOCEAN UNIVERSITY

```
# Set the initial speed of the motors to 10% of the maximum speed
leftMotor.setVelocity(0.1 * MAX_SPEED)
rightMotor.setVelocity(0.1 * MAX_SPEED)
# ضبط السرعة الأولية للمحركات إلى 10% من السرعة القصوى


 #Main loop: run until the simulation is terminated
while robot.step(TIME_STEP) != -1:
    # In this loop, we don't need to do anything because the robot will keep moving forward
    pass
# الحلقة الرئيسية: تستمر حتى يتم إنهاء المحاكاة
# في هذه الحلقة، لا نحتاج إلى فعل أي شيء لأن الروبوت سيستمر في التحرك للأمام
```

EXAMPLES 2

Moving the e-puck robot **forward** for **10 steps** and then **turning right**

epuck_move_turn.py



Day
01

MIDOCEAN UNIVERSITY

1. **Importing Classes:**

   - `` `from controller import Robot, Motor` ``: Imports the `` `Robot` `` and `` `Motor` `` classes from the Webots controller module.
   - `` `from controller import Robot, Motor` ``: يستورد الفئات `` `Robot` `` و `` `Motor` `` من وحدة التحكم Webots.

2. **Defining Constants:**

   - `` `TIME_STEP = 64` ``: Defines the time step for the simulation. This should match the time step set in Webots.
   - `` `TIME_STEP = 64` ``: يحدد خطوة الزمن للمحاكاة. يجب أن تطابق هذه القيمة خطوة الوقت في Webots.
   - `` `MAX_SPEED = 6.28` ``: Defines the maximum speed for the e-puck robot's motors.
   - `` `MAX_SPEED = 6.28` ``: يحدد السرعة القصوى لمحركات الروبوت e-puck.

MIDOCEAN UNIVERSITY

3. Creating the Robot Instance:

- `robot = Robot()`: Creates an instance of the `Robot` class, allowing you to control the e-puck robot.
- `robot = Robot()`: ينشئ مثيلًا لفئة `Robot`، مما يتيح لك التحكم في الروبوت e-puck.

4. Getting the Motors:

- `leftMotor = robot.getDevice('left wheel motor')`: Gets the left wheel motor.
- `leftMotor = robot.getDevice('left wheel motor')`: يحصل على محرك العجلة اليسرى.
- `rightMotor = robot.getDevice('right wheel motor')`: Gets the right wheel motor.
- `rightMotor = robot.getDevice('right wheel motor')`: يحصل على محرك العجلة اليمنى.

MIDOCEAN UNIVERSITY

5. Setting Motor Positions:

- `leftMotor.setPosition(float('inf'))`: Sets the left motor position to infinity, enabling speed control.
- `leftMotor.setPosition(float('inf'))`: يضبط موضع المحرك الأيسر إلى ما لا نهاية، مما يتيح التحكم في السرعة.
- `rightMotor.setPosition(float('inf'))`: Sets the right motor position to infinity, enabling speed control.
- `rightMotor.setPosition(float('inf'))`: يضبط موضع المحرك الأيمن إلى ما لا نهاية، مما يتيح التحكم في السرعة.

6. Moving Forward:

- The loop `for step in range(10)` runs for 10 iterations, moving the robot forward by setting both wheel speeds to 50% of the maximum speed.
- الحلقة `for step in range(10)` تستمر لـ 10 تكرارات، تحرك الروبوت للأمام عن طريق ضبط سرعتي العجلتين إلى 50% من السرعة القصوى.

7. **Turning Right:**

   - The loop `` `for step in range(10)` `` runs for 10 iterations, turning the robot right by setting the left wheel speed to 50% of the maximum speed and the right wheel speed to 0.
   - الحلقة `` `for step in range(10)` `` تستمر لـ 10 تكرارات، تدور الروبوت إلى اليمين عن طريق ضبط سرعة العجلة اليسرى إلى 50% من السرعة القصوى وسرعة العجلة اليمنى إلى الصفر.

8. **Stopping the Robot:**

   - The loop `` `while robot.step(TIME_STEP) != -1` `` runs indefinitely, keeping the robot stopped by setting both wheel speeds to 0.
   - الحلقة `` `while robot.step(TIME_STEP) != -1` `` تستمر بلا نهاية، تحافظ على الروبوت متوقفًا عن طريق ضبط سرعتي العجلتين إلى الصفر.

Day
01

MIDOCEAN UNIVERSITY

Day
02

MIDOCEAN
UNIVERSITY

REVISION

Webots

robot simulation

MIDOCEAN UNIVERSITY

Tutorial 1: Moving the Robot Forward

Objective: Learn how to initialize your robot and make it move forward.

Step 1: Setting Up Your Webots Project

- Open Webots and create a new project.
- Add an e-puck robot to your scene from the Webots library.

Step 2: Creating Your Controller

- Navigate to **File** > **New** > **New Robot Controller...**.
- Name your controller **move_forward**.
- Select Python as the programming language and confirm.

Step 3: Writing the Controller Code

Open the **move_forward.py** file generated by Webots and follow along:

Day

02

MIDOCEAN UNIVERSITY

```python
# Import the Robot class to interact with the robot
from controller import Robot

# Create a robot instance to control it
robot = Robot()

# Define the simulation step time
TIME_STEP = 64

# Main control loop
while robot.step(TIME_STEP) != -1:
    # This loop runs until the simulation is stopped
    pass  # We will add movement code in the next tutorial
```

This code initializes the robot but doesn't yet move it. Let's proceed to add movement.

```python
# Import the Motor class to control the wheels
from controller import Robot, Motor

# Initialization steps as before

# Retrieve motor devices
leftMotor = robot.getDevice('left wheel motor')
rightMotor = robot.getDevice('right wheel motor')
# Set the motors to velocity control mode
leftMotor.setPosition(float('inf'))
rightMotor.setPosition(float('inf'))
# Define a forward speed
FORWARD_SPEED = 6.28
# Apply the speed to both motors to move forward
leftMotor.setVelocity(FORWARD_SPEED)
rightMotor.setVelocity(FORWARD_SPEED)
```

```python
# Main control loop
while robot.step(TIME_STEP) != -1:
    # This loop runs until the simulation is stopped
    pass  # We will add movement code in the next tutorial
```

Day
02

Example 1:

```
# Import the Robot class to interact with the robot
from controller import Robot

# Create a robot instance to control it
robot = Robot()

# Define the simulation step time
TIME_STEP = 64

# Main control loop
while robot.step(TIME_STEP) != -1:
    # This loop runs until the simulation is stopped
    pass  # We will add movement code in the next tutorial
```

Example 2:

```
# Import the Motor class to control the wheels
from controller import Robot, Motor

# Initialization steps as before

# Retrieve motor devices
leftMotor = robot.getDevice('left wheel motor')
rightMotor = robot.getDevice('right wheel motor')
# Set the motors to velocity control mode
leftMotor.setPosition(float('inf'))
rightMotor.setPosition(float('inf'))
# Define a forward speed
FORWARD_SPEED = 6.28
# Apply the speed to both motors to move forward
leftMotor.setVelocity(FORWARD_SPEED)
rightMotor.setVelocity(FORWARD_SPEED)

# Main control loop
while robot.step(TIME_STEP) != -1:
    # This loop runs until the simulation is stopped
    pass  # We will add movement code in the next tutorial
```

Day
02

MIDOCEAN UNIVERSITY

OBSTACLE AVOIDANCE ROBOT USING IR SENSORS

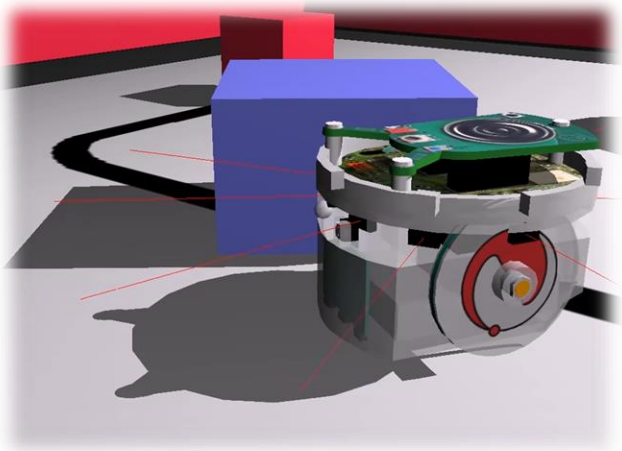
Tutorial 2: Avoiding Obstacles Using a Front Infrared Sensor

Example3_epuck_avoid obstacle_using_ps0

Objective: Modify the robot's behavior to stop when an obstacle is detected.

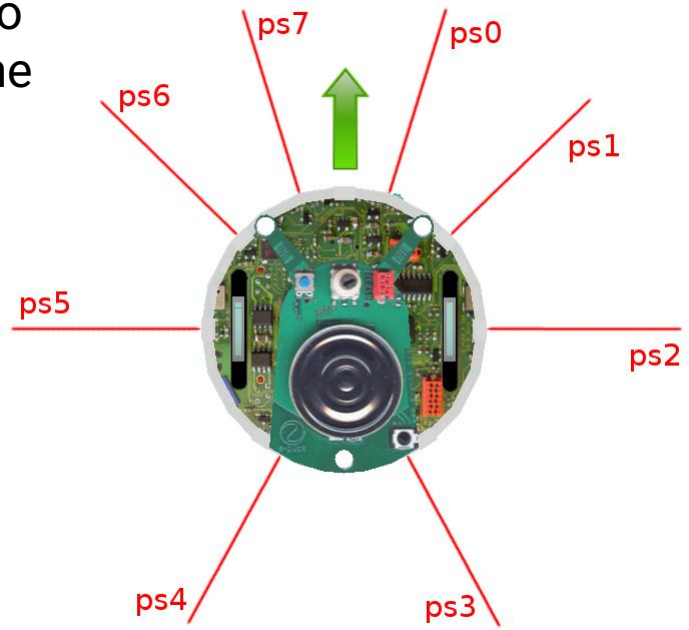
Step 1: Enhancements to the Controller

Build upon the **move_forward** controller by adding sensor functionality to detect obstacles.



MIDOCEAN UNIVERSITY

Controller programming requires some information related to the e-puck model. In order to create the collision avoidance algorithm, we need to read the values of its 8 infra-red distance sensors located around its turret, and we need to actuate its two wheels. The way that the distance sensors are distributed around the turret and the e-puck direction are depicted in this figure.



Day
02

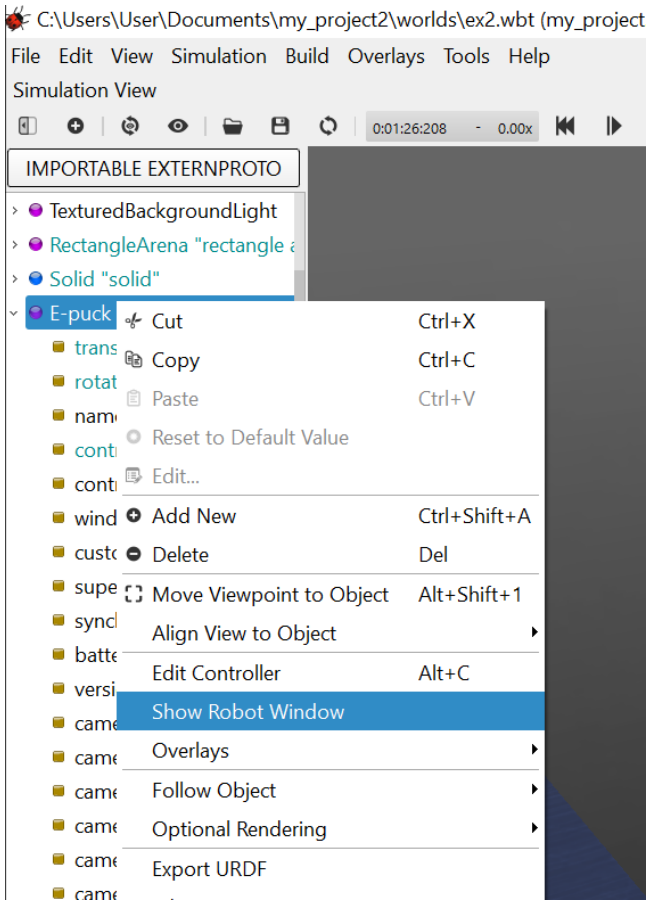
MIDOCEAN UNIVERSITY

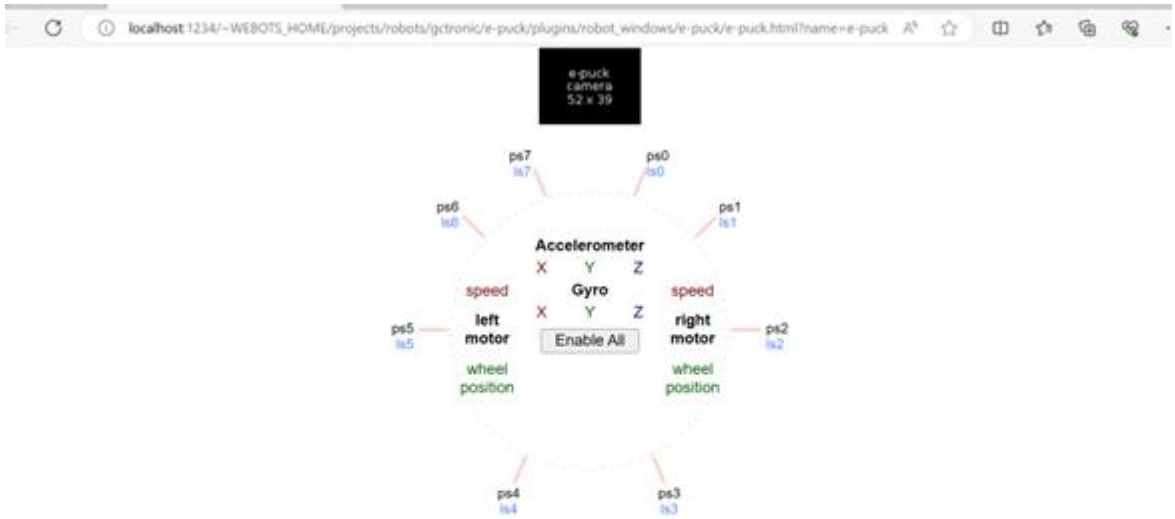
```
# Import necessary classes
from controller import Robot, Motor, DistanceSensor

# Initialization steps as before

# Initialize the front distance sensor
frontSensor = robot.getDevice('ps0')
frontSensor.enable(TIME_STEP)

# Control loop
while robot.step(TIME_STEP) != -1:
    # Read the sensor value
    distance = frontSensor.getValue()

    # Check for obstacles
    if distance < 800:  # Adjust based on your environment
        # An obstacle is detected, stop the robot
        leftMotor.setVelocity(0)
        rightMotor.setVelocity(0)
    else:
        # No obstacle, move forward
        leftMotor.setVelocity(FORWARD_SPEED)
        rightMotor.setVelocity(FORWARD_SPEED)
```

Day

02

MIDOCEAN UNIVERSITY





```
from controller import Robot, Motor, DistanceSensor

TIME_STEP = 64
MAX_SPEED = 6.28  # Adjusting MAX_SPEED for realism, assuming it's in radians/sec which is typical in simulations

# create the Robot instance.
robot = Robot()

# get a handler to the motors and set target position to infinity (speed control)
leftMotor = robot.getDevice('left wheel motor')
rightMotor = robot.getDevice('right wheel motor')
leftMotor.setPosition(float('inf'))
rightMotor.setPosition(float('inf'))

# Initialize and enable the distance sensor
distanceSensor = robot.getDevice('ps0')  # Assuming 'sp0' is the correct sensor name
distanceSensor.enable(TIME_STEP)

# set up the motor speeds at 10% of the MAX_SPEED.
leftMotor.setVelocity(0.1 * MAX_SPEED)
rightMotor.setVelocity(0.1 * MAX_SPEED)

while robot.step(TIME_STEP) != -1:
    # Read the sensor value
    distance = distanceSensor.getValue()
    print(distance)

    # Check for obstacles - assuming a threshold value for when to stop
    # You may need to adjust the threshold based on your simulation environment
    if distance < 62:  # Example threshold value, adjust based on your sensor's range and sensitivity
        # An obstacle is detected, stop the robot
        leftMotor.setVelocity(MAX_SPEED)
        rightMotor.setVelocity(0)
    else:
        # No obstacle, continue moving forward
        leftMotor.setVelocity(0.1 * MAX_SPEED)
        rightMotor.setVelocity(0.1 * MAX_SPEED)
```

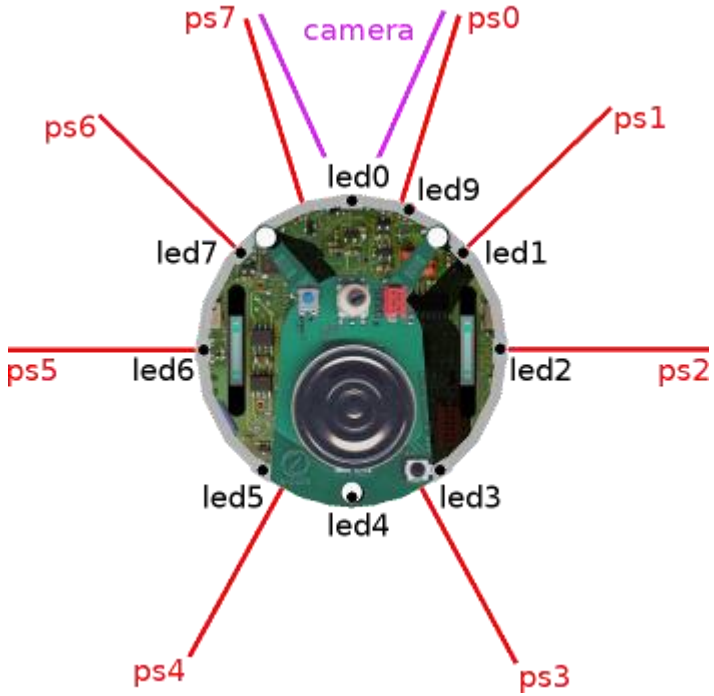
Day
02

Multiple Infrared sensors

MIDOCEAN UNIVERSITY

Day
02

e-puck robot



Tutorial 3: Using Multiple Infrared Sensors for Enhanced Obstacle Avoidance

Example4_epuck_avoid obstacle_using_multiple_infrered

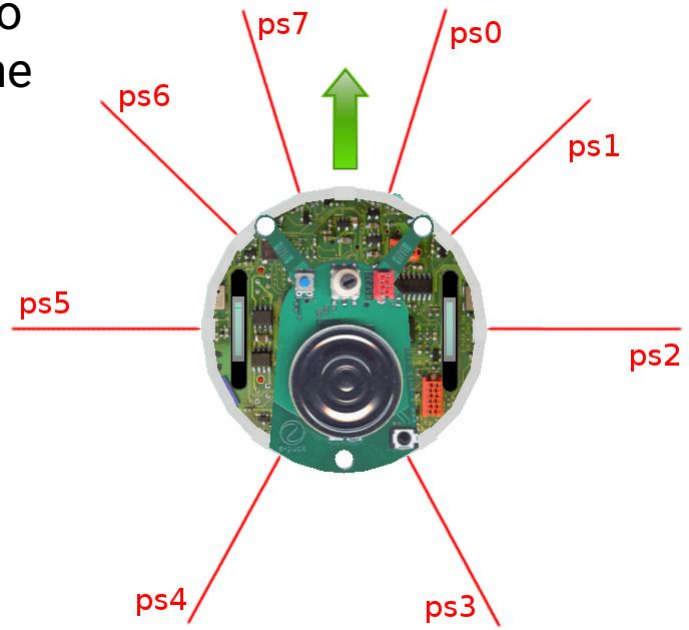
Objective: Expand obstacle detection to utilize **multiple infrared sensors** for more nuanced navigation.

Step 1: Controller Expansion for Multiple Sensors

Enhance the controller to read from multiple sensors and react accordingly.

MIDOCEAN UNIVERSITY

Controller programming requires some information related to the e-puck model. In order to create the collision avoidance algorithm, we need to read the values of its 8 infra-red distance sensors located around its turret, and we need to actuate its two wheels. The way that the distance sensors are distributed around the turret and the e-puck direction are depicted in this figure.



Front: ps0 and ps7

Front-right: ps1

Front-left: ps6

Obstacle detection and navigation

1. **Sensors for Obstacle Detection in Front:**
    - `ps0` and `ps7` are used to detect obstacles directly in front of the robot.
2. **Control Loop:**
    - If `ps0` or `ps7` detects an obstacle (sensor value less than 70), the robot stops.
    - If `ps1` detects an obstacle (sensor value less than 70), the robot turns left.
    - If `ps6` detects an obstacle (sensor value less than 70), the robot turns right.
    - If no obstacle is detected, the robot moves forward.

```python
# Import necessary classes for the robot control
from controller import Robot, Motor, DistanceSensor

# Initialize the Robot instance
robot = Robot()

# Define the time step of the robot's control loop
TIME_STEP = 64

# Initialize motor devices
leftMotor = robot.getDevice('left wheel motor')
rightMotor = robot.getDevice('right wheel motor')

# Set to infinite position mode
leftMotor.setPosition(float('inf'))
rightMotor.setPosition(float('inf'))

# Initial velocity
leftMotor.setVelocity(0.0)
rightMotor.setVelocity(0.0)
```

```python
# Speed constants
FORWARD_SPEED = 5.0

# Initialize multiple distance sensors by their names
sensors = ['ps0', 'ps7', 'ps1', 'ps6']
sensorObjects = []

for sensorName in sensors:
    sensor = robot.getDevice(sensorName)  # Get the sensor device
    sensor.enable(TIME_STEP)  # Enable the sensor with the defined time step
    sensorObjects.append(sensor)  # Append the sensor object to the list
```

```python
# Control loop: execute at each simulation step
while robot.step(TIME_STEP) != -1:
    # Read sensor values into a list
    sensorValues = [sensor.getValue() for sensor in sensorObjects]

    # Basic obstacle detection and navigation strategy
    if sensorValues[0] < 70 or sensorValues[1] < 70:  # ps0 or ps7 detects an obstacle in fr
        # Obstacle detected in front: stop the robot
        leftMotor.setVelocity(0)
        rightMotor.setVelocity(0)
    elif sensorValues[2] < 70:  # ps1 detects an obstacle in the front-right
        # Obstacle detected in front-right: turn left
        leftMotor.setVelocity(-FORWARD_SPEED)
        rightMotor.setVelocity(FORWARD_SPEED)
    elif sensorValues[3] < 70:  # ps6 detects an obstacle in the front-left
        # Obstacle detected in front-left: turn right
        leftMotor.setVelocity(FORWARD_SPEED)
        rightMotor.setVelocity(-FORWARD_SPEED)
    else:
        # No obstacle detected: move forward
        leftMotor.setVelocity(FORWARD_SPEED)
        rightMotor.setVelocity(FORWARD_SPEED)
```

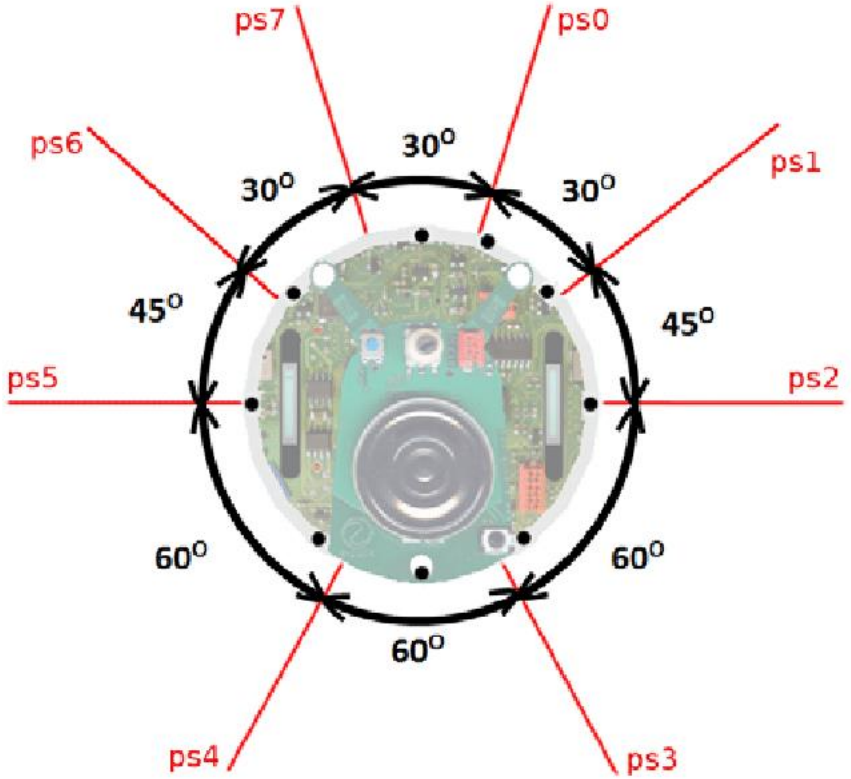
MIDOCEAN UNIVERSITY

e-puck Robot Sensor Positions

The e-puck robot has 8 distance sensors placed around its body. These sensors can detect obstacles at different angles around the robot. Here's a breakdown of the sensors and their positions:

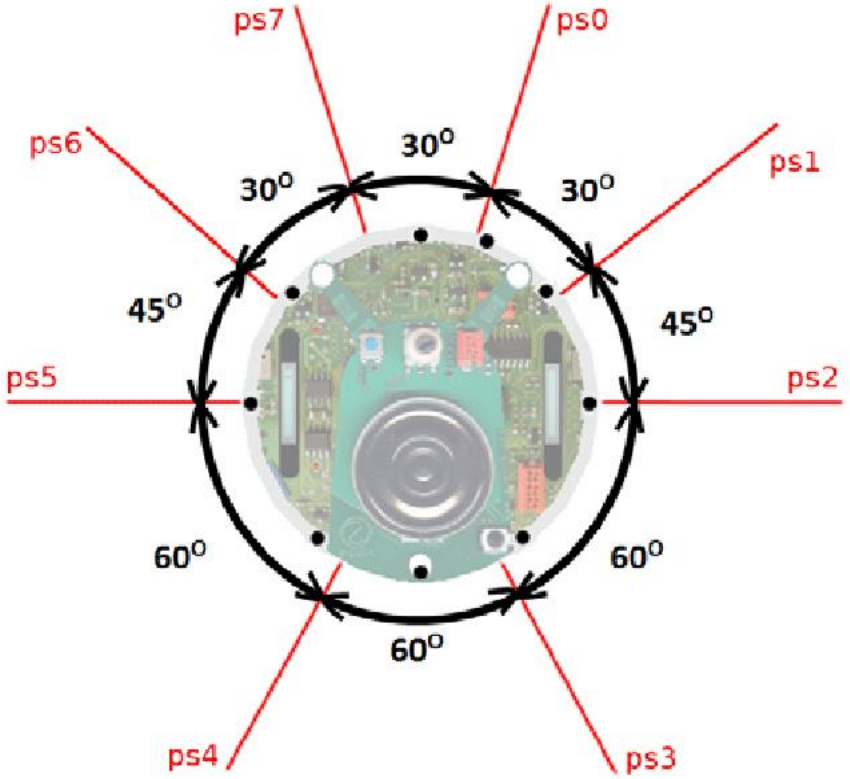
- **ps0**: Front-center
- **ps1**: Front-right
- **ps2**: Right
- **ps3**: Back-right
- **ps4**: Back-center
- **ps5**: Back-left
- **ps6**: Left
- **ps7**: Front-left

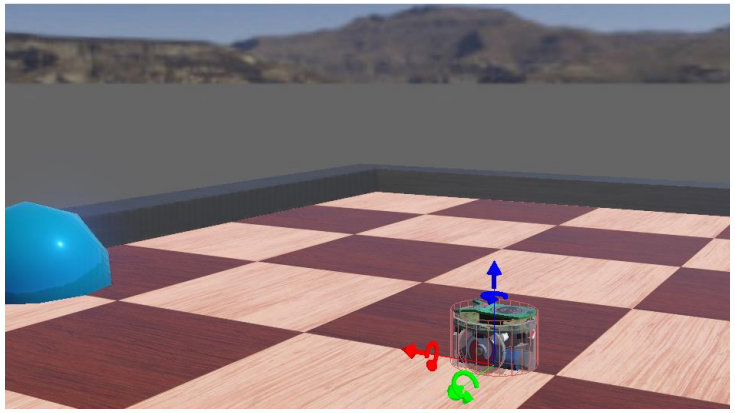
The sensors are arranged in a circular pattern, allowing the robot to detect obstacles in almost all directions.



EXAMPLES 5

Robot that flashes its LED red and white three times before turning right when it detects an obstacle.





Example5_epuck_obstacle_detection_led

```python
# Import necessary classes for the robot control
from controller import Robot, Motor, DistanceSensor, LED

# Initialize the Robot instance
robot = Robot()

# Define the time step of the robot's control loop
TIME_STEP = 64

# Initialize motor devices
leftMotor = robot.getDevice('left wheel motor')
rightMotor = robot.getDevice('right wheel motor')

# Set to infinite position mode
leftMotor.setPosition(float('inf'))
rightMotor.setPosition(float('inf'))

# Initial velocity
leftMotor.setVelocity(0.0)
rightMotor.setVelocity(0.0)
```

```python
# Speed constants
FORWARD_SPEED = 5.0

# Initialize multiple distance sensors by their names
sensors = ['ps0', 'ps7', 'ps1', 'ps6']
sensorObjects = []

for sensorName in sensors:
    sensor = robot.getDevice(sensorName)  # Get the sensor device
    sensor.enable(TIME_STEP)  # Enable the sensor with the defined time step
    sensorObjects.append(sensor)  # Append the sensor object to the list

# Initialize LED device (assuming LED 0 is used)
led = robot.getDevice('led0')

# Function to flash the LED red and white three times
def flashLED():
    for _ in range(3):
        led.set(1)  # Turn LED red
        robot.step(TIME_STEP)
        led.set(0)  # Turn LED white
        robot.step(TIME_STEP)
```

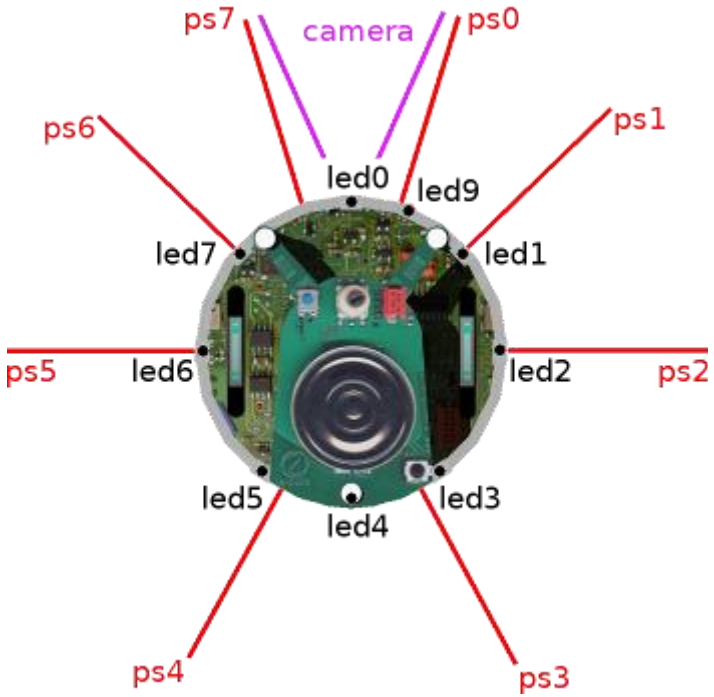
```
# Control loop: execute at each simulation step
while robot.step(TIME_STEP) != -1:
    # Read sensor values into a list
    sensorValues = [sensor.getValue() for sensor in sensorObjects]

    # Basic obstacle detection and navigation strategy
    if sensorValues[0] < 70 or sensorValues[1] < 70:  # ps0 or ps7 detects an obstacle in fr
        # Obstacle detected in front: flash LED and then turn right
        flashLED()
        leftMotor.setVelocity(FORWARD_SPEED)
        rightMotor.setVelocity(-FORWARD_SPEED)
    else:
        # No obstacle detected: move forward
        leftMotor.setVelocity(FORWARD_SPEED)
        rightMotor.setVelocity(FORWARD_SPEED)
```

Day
02

Camera Sensor

MIDOCEAN UNIVERSITY

E-puck robot can use its **camera** to capture an image and analyze the color of the central pixel in the Webots simulation environment.



Example6_epuck_camera_capture

1- Import Necessary Modules:
Imports the necessary classes from the Webots controller module.

```
from controller import Robot, Camera
```

2-Define the Time Step:
Sets the time step for the control loop.

```
TIME_STEP = 64
```

3- Initialize the Robot:
Creates an instance of the Robot class, which will be used to interact with the robot.

```
robot = Robot()
```

4-Get and Enable the Camera:
Gets the camera device and enables it to start capturing images.

```
camera = robot.getDevice('camera')
camera.enable(TIME_STEP)
```

5- Main Control Loop:

- Captures images, retrieves their dimensions, and prints the size.
- Calculates the central pixel position and retrieves its RGB values.
- Prints the RGB values of the central pixel for analysis or detection tasks.

```
while robot.step(TIME_STEP) != -1:
    image = camera.getImage()
    cameraWidth = camera.getWidth()
    cameraHeight = camera.getHeight()
    print(f"Captured an image of size: {cameraWidth}x{cameraHeight}")

    centerX = cameraWidth // 2
    centerY = cameraHeight // 2

    red = camera.imageGetRed(image, cameraWidth, centerX, centerY)
    green = camera.imageGetGreen(image, cameraWidth, centerX, centerY)
    blue = camera.imageGetBlue(image, cameraWidth, centerX, centerY)

    print(f"Color of the central pixel: R={red}, G={green}, B={blue}")
```

```
# Import necessary modules from the Webots library.
# Robot is the main class for robot control.
# Camera is for accessing the robot's camera.
from controller import Robot, Camera

# Define the simulation time step in milliseconds.
# This is how often the simulation updates the robot's sensors and actuators.
TIME_STEP = 64

# Create an instance of the Robot class.
# This will be our main handle to interact with the robot.
robot = Robot()

# Get a handle to the camera device on the robot.
# 'camera' is the name of the camera device specified in the robot's model.
# If the camera has a different name in your project, change 'camera' accordingly.
camera = robot.getDevice('camera')

# Enable the camera to start capturing images.
# The camera will update its image every TIME_STEP milliseconds.
camera.enable(TIME_STEP)

# Main loop: Execute actions repeatedly until the simulation is stopped.
while robot.step(TIME_STEP) != -1:
    # Capture an image from the camera.
    # The getImage() method returns the latest image captured by the camera.
    image = camera.getImage()
    # Retrieve the width and height of the camera image.
    # These values are set in the Webots world file and can be used to process the image.
    cameraWidth = camera.getWidth()
    cameraHeight = camera.getHeight()
    print(f"Captured an image of size: {cameraWidth}x{cameraHeight}")

    # Calculate the position of the central pixel in the image.
    # We divide the width and height by 2 to find the center.
    centerX = cameraWidth // 2
    centerY = cameraHeight // 2

    # Get the RGB color values of the central pixel.
    # imageGetRed, imageGetGreen, and imageGetBlue methods are used to extract
    # the color components of a pixel at a specified position.
    red = camera.imageGetRed(image, cameraWidth, centerX, centerY)
    green = camera.imageGetGreen(image, cameraWidth, centerX, centerY)
    blue = camera.imageGetBlue(image, cameraWidth, centerX, centerY)

    # Print the RGB color values of the central pixel.
    # This can be useful for tasks like color detection or analysis.
    print(f"Color of the central pixel: R={red}, G={green}, B={blue}")
```

Day

02

Obstacle

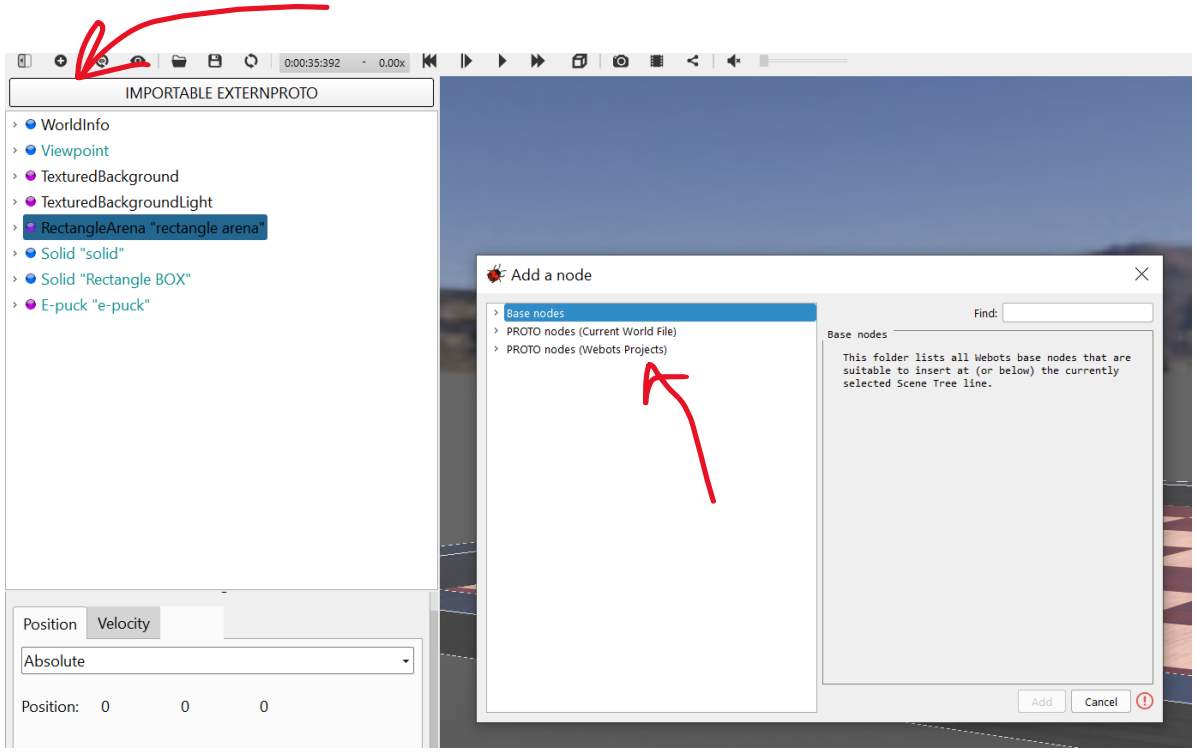
MIDOCEAN UNIVERSITY

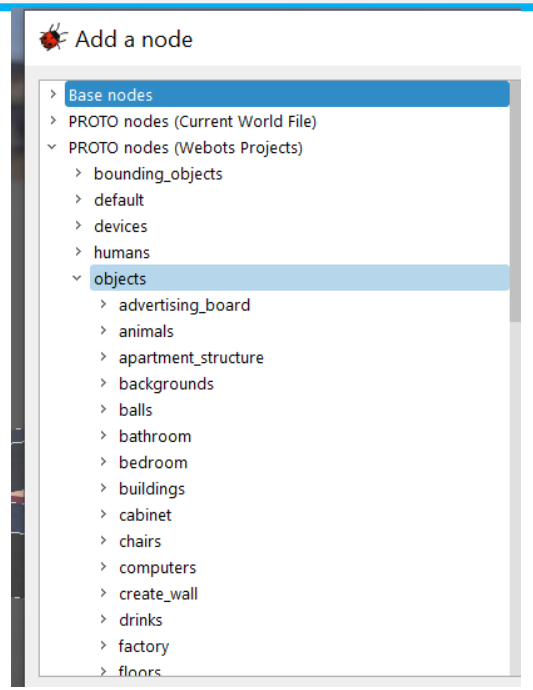
Day

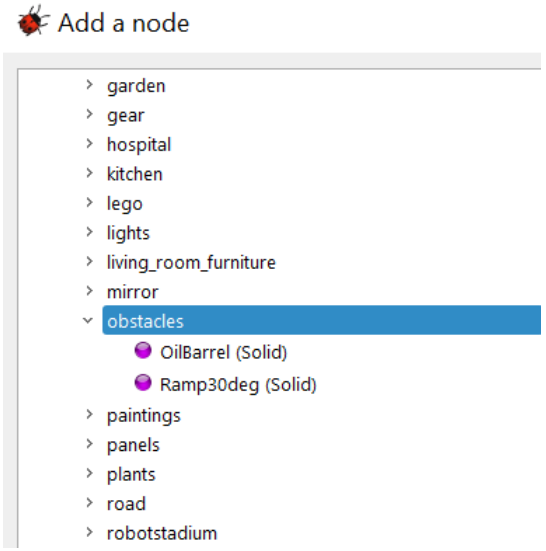
02

ADD Obstacle (ROCK) in Webots

MIDOCEAN UNIVERSITY





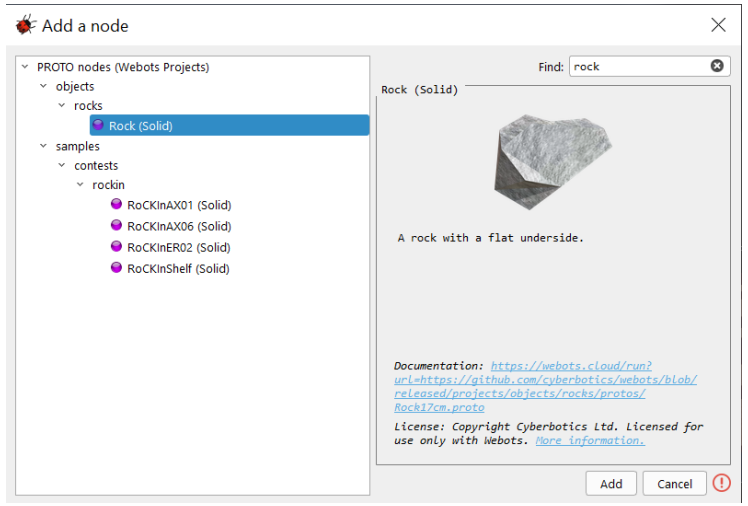


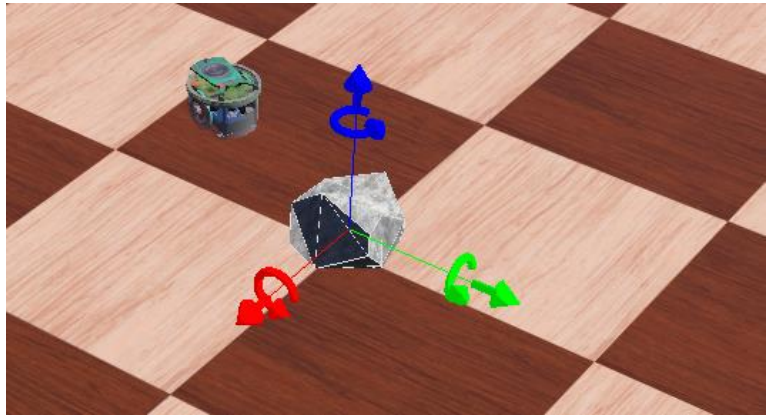
\+ > Add a node > PROTO nodes>Objects > search rocks >

Day

02

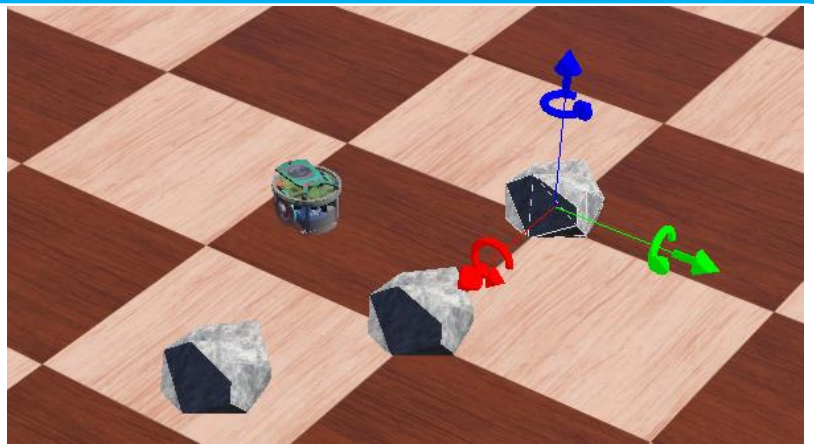
MIDOCEAN UNIVERSITY



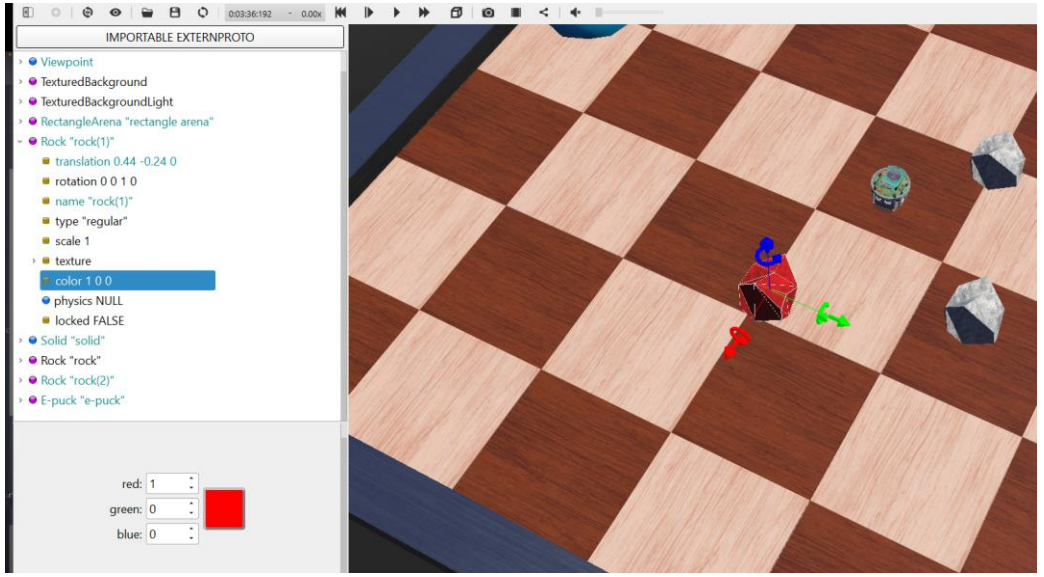


Click on the Arrows to move the object

Ctrl + V to copy paste the object



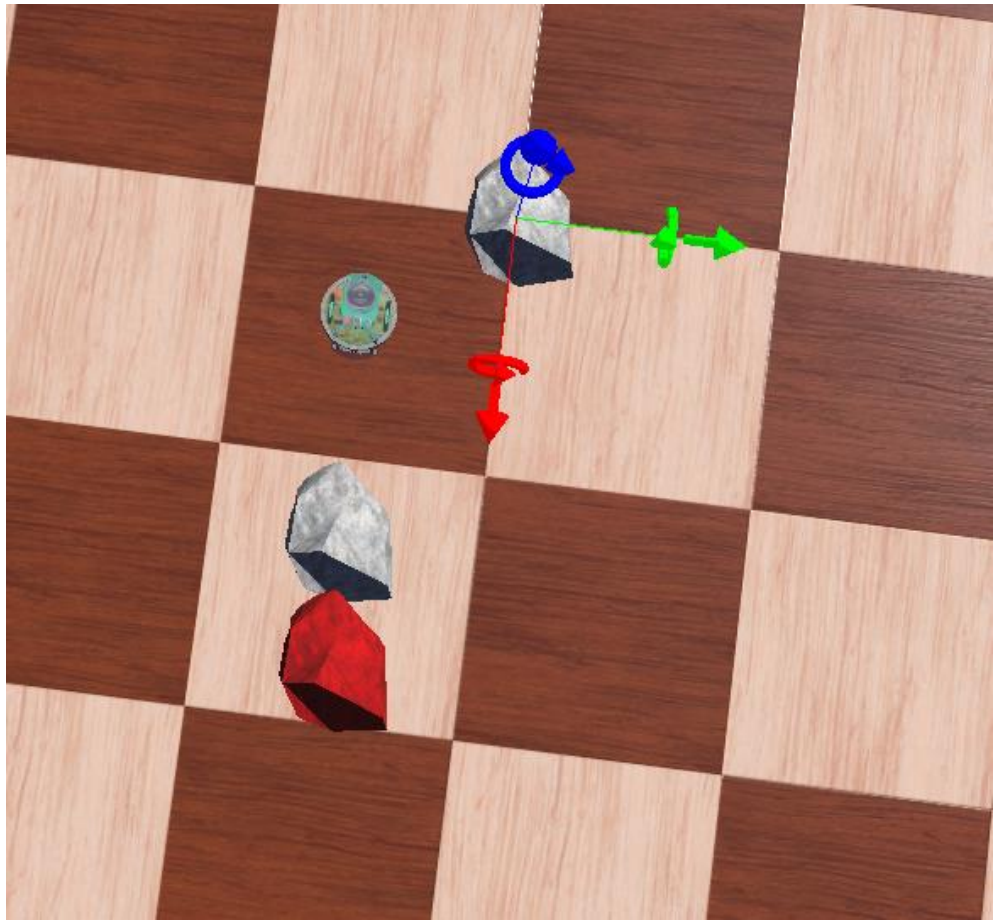
Change the color of Rock



Day
02

Epuck Camera with Color Object

MIDOCEAN UNIVERSITY

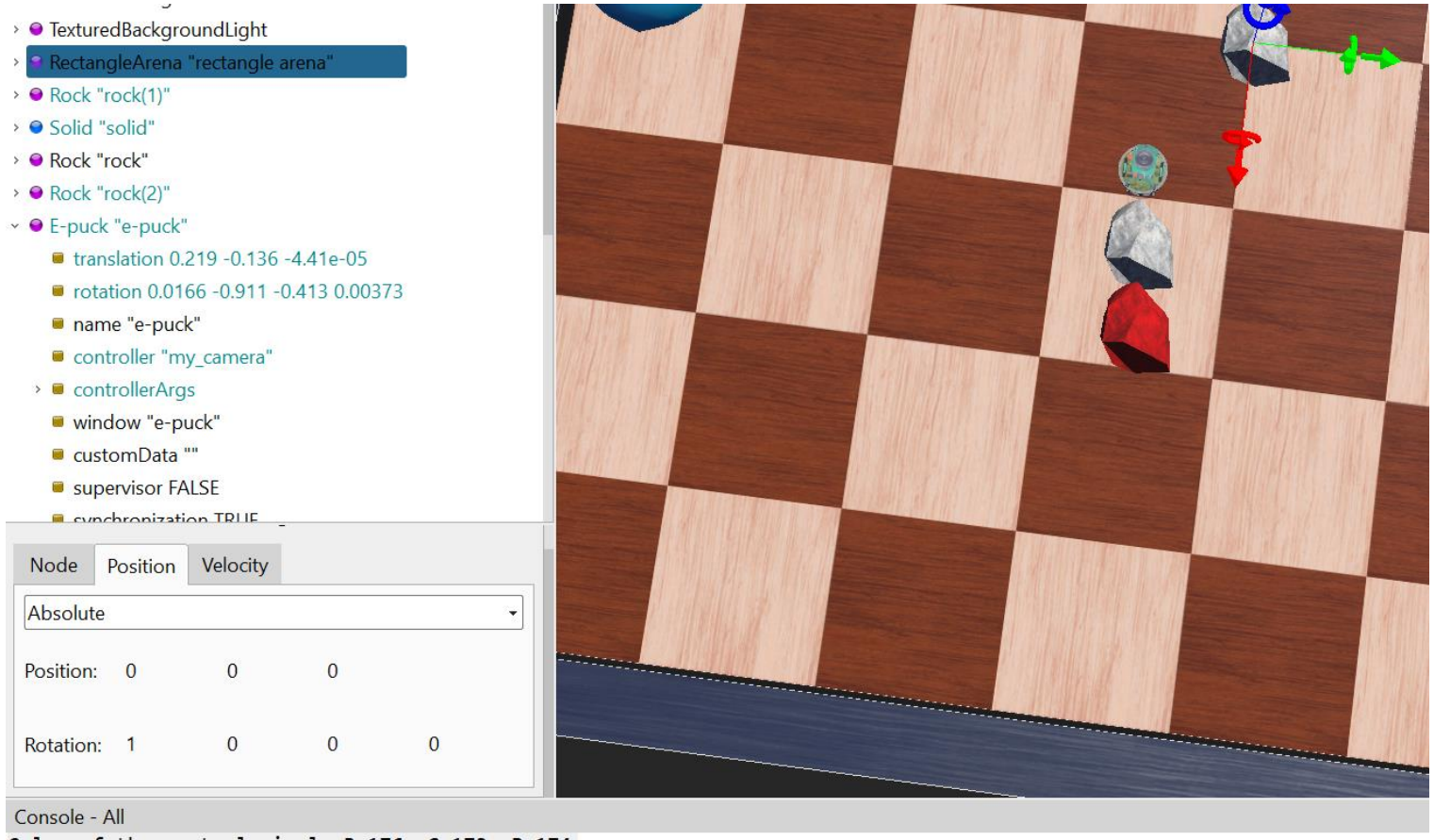


1- Change the Location of the Gray Object to be in front of the Epuck Robot.

2- Read the RGB Values of the center of Camera

```
from controller import Robot, Camera

# Time step definition
TIME_STEP = 64

# Create the Robot instance
robot = Robot()

# Initialize the camera and enable it
camera = robot.getDevice('camera')
camera.enable(TIME_STEP)

# Main control loop
while robot.step(TIME_STEP) != -1:
    # Capture an image from the camera
    image = camera.getImage()

    # Assuming you've set the resolution in the Webots world file
    cameraWidth = camera.getWidth()
    cameraHeight = camera.getHeight()
    print(f"Captured an image of size: {cameraWidth}x{cameraHeight}")

    # Get the color of the central pixel
    centerX = cameraWidth // 2
    centerY = cameraHeight // 2
    red = camera.imageGetRed(image, cameraWidth, centerX, centerY)
    green = camera.imageGetGreen(image, cameraWidth, centerX, centerY)
    blue = camera.imageGetBlue(image, cameraWidth, centerX, centerY)

    print(f"Color of the central pixel: R={red}, G={green}, B={blue}")
```

MIDOCEAN UNIVERSITY



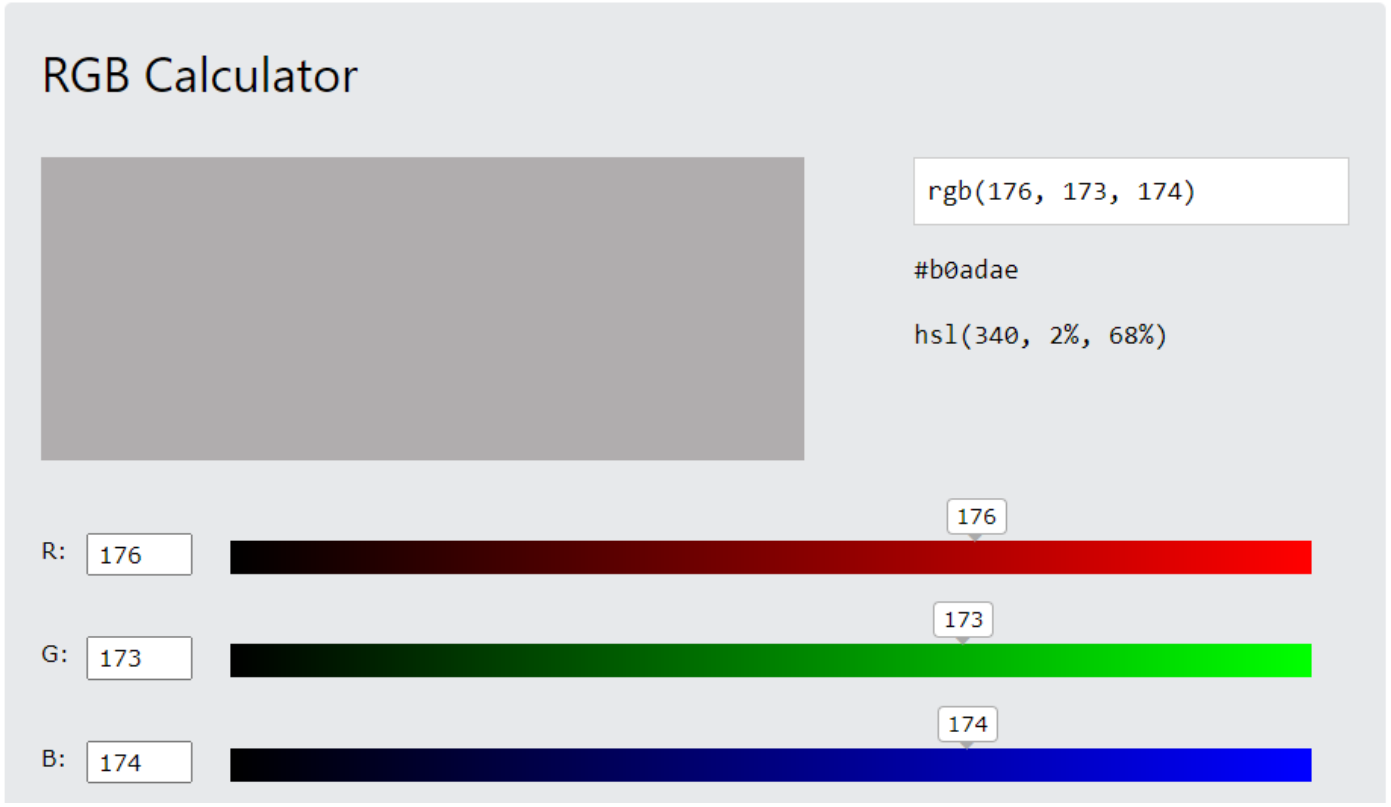
The output value of RGB:
R=176 , G=173, B=174

Use any website to convert RGB values to Color:
https://www.w3schools.com/colors/colors_rgb.asp

Day

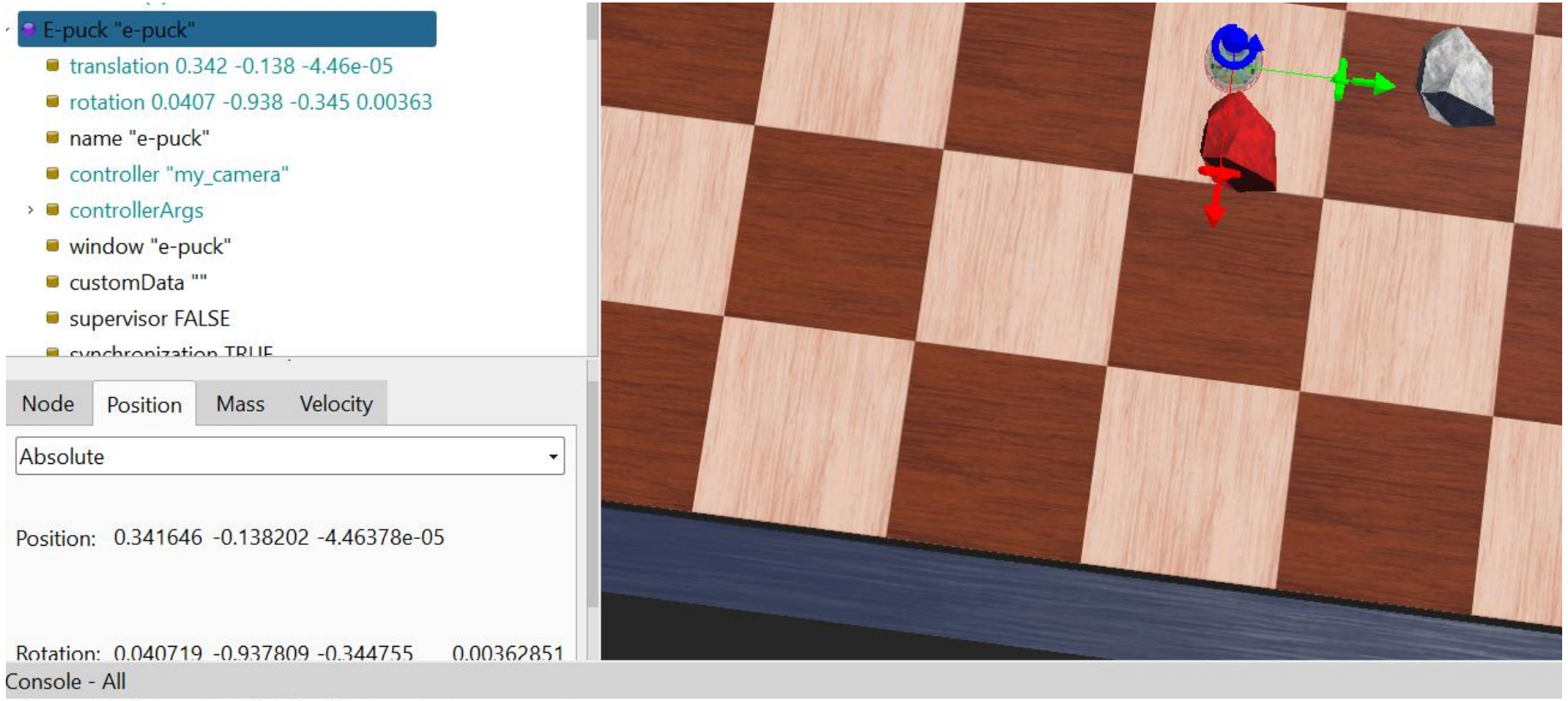
02

MIDOCEAN UNIVERSITY



MIDOCEAN UNIVERSITY

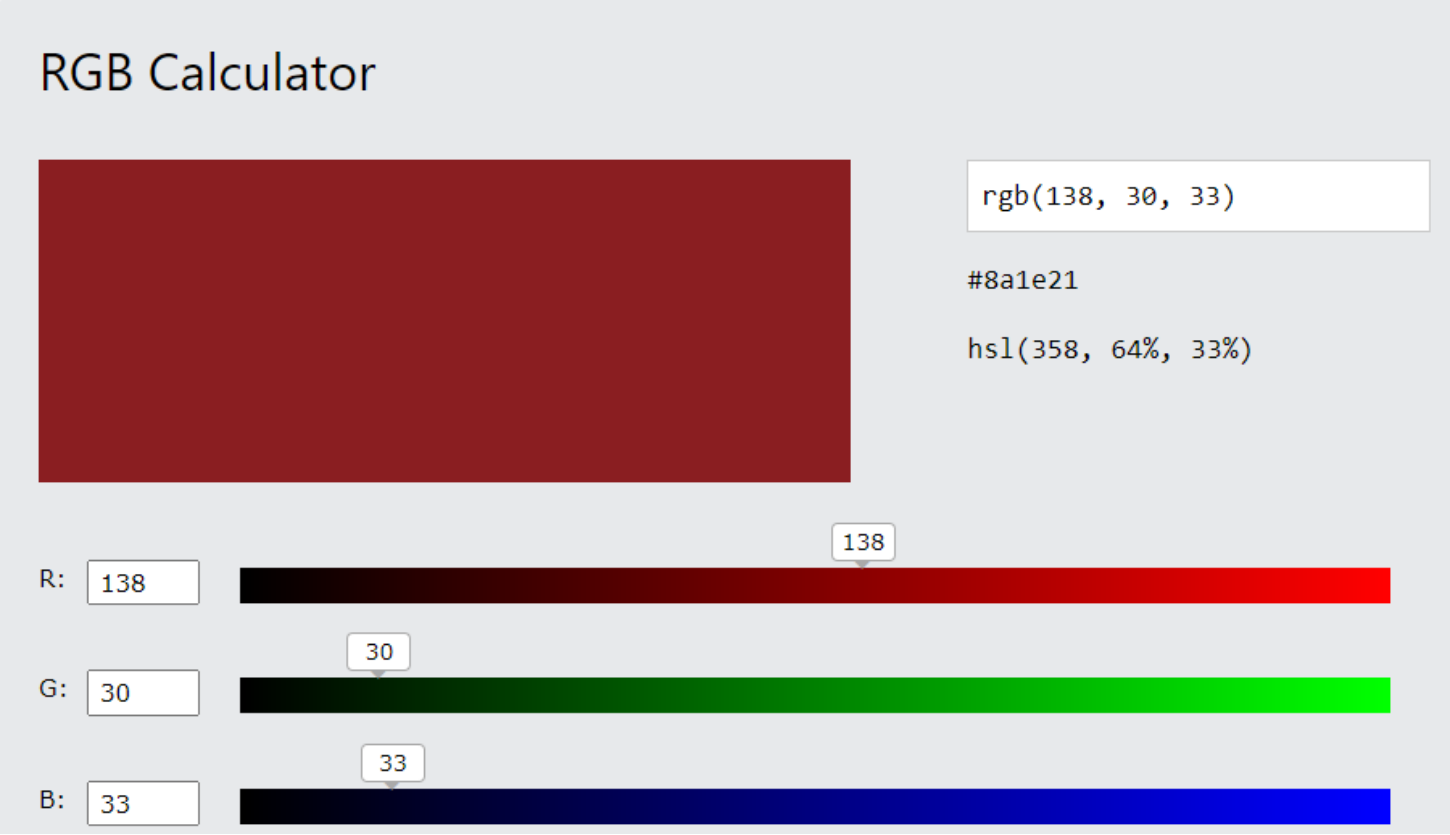
Now Remove the gray object and let Epuck robot camera capture the red object



Day

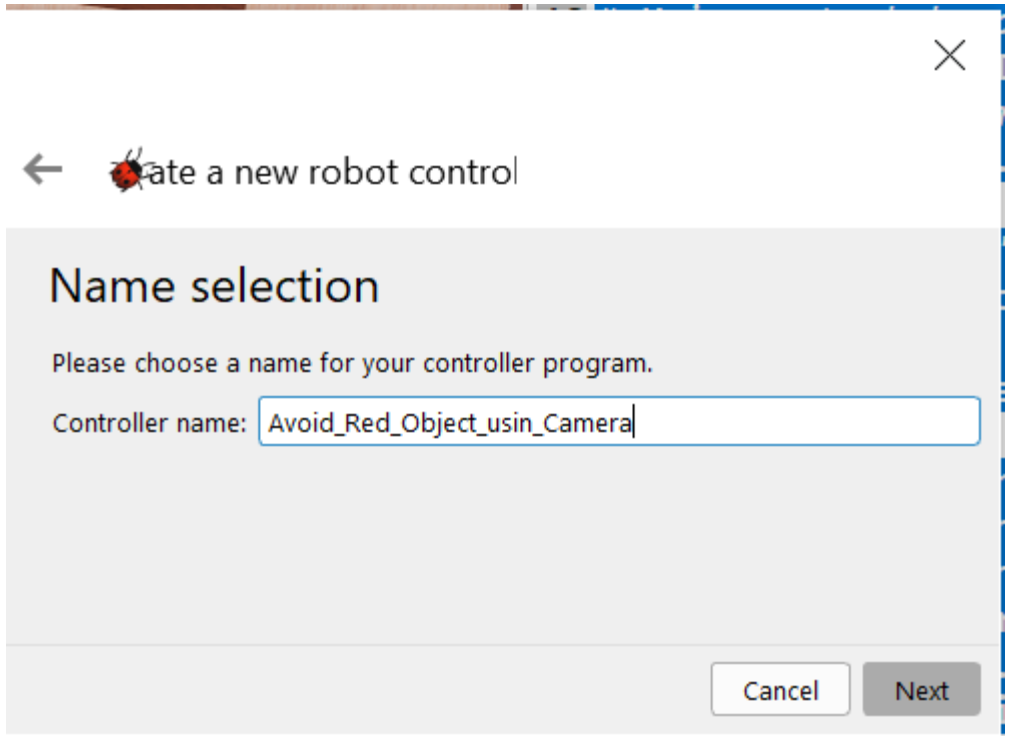
02

MIDOCEAN UNIVERSITY



Example7_epuck_red_obstacle_detection

Epuck robot avoid RED obstacle using Camera



- **Move Forward**:
  - The robot will move forward when the central pixel is not sufficiently red (i.e., it does not meet the red threshold criteria).
  - The left and right motors are set to a positive velocity (`5`), making the robot move forward.
- **Turn Right**:
  - The robot will turn right when the central pixel is sufficiently red (i.e., it meets the red threshold criteria).
  - The left motor is set to a positive velocity (`2`), and the right motor is set to a negative velocity (`-2`), making the robot turn right.

The robot will turn right when the central pixel in the camera's view is identified as "red enough." This determination is made using the following conditions:

1. The red component (`red`) of the central pixel must be **greater than** `RED_THRESHOLD`.
2. The green component (`green`) of the central pixel must be **less than** `GREEN_THRESHOLD`.
3. The blue component (`blue`) of the central pixel must be **less than** `BLUE_THRESHOLD`.

```
# Import the necessary modules from the Webots API
from controller import Robot, Camera, Motor

# Set the time step of the simulation
TIME_STEP = 64

# Create a robot instance to interact with the robot
robot = Robot()

# Get the camera device, enable it, and set the timestep
camera = robot.getDevice('camera')
camera.enable(TIME_STEP)

# Get the motor devices
left_motor = robot.getDevice('left wheel motor')
right_motor = robot.getDevice('right wheel motor')

# Set the target position of the motors to infinity for velocity control
left_motor.setPosition(float('inf'))
right_motor.setPosition(float('inf'))

# Set the initial velocity of the motors to 0 (robot is stationary)
left_motor.setVelocity(0)
right_motor.setVelocity(0)
```

```
# Main loop: this is where the robot's behavior is defined
while robot.step(TIME_STEP) != -1:
    # Retrieve the image captured by the camera
    image = camera.getImage()

    # Get the width and height of the image from the camera's properties
    width = camera.getWidth()
    height = camera.getHeight()

    # Calculate the central pixel's coordinates
    centerX = width // 2
    centerY = height // 2

    # Get the red, green, and blue color components of the central pixel
    red = camera.imageGetRed(image, width, centerX, centerY)
    green = camera.imageGetGreen(image, width, centerX, centerY)
    blue = camera.imageGetBlue(image, width, centerX, centerY)
    print(f"Color of the central pixel: R={red}, G={green}, B={blue}")
```

```
    # Define what we mean by 'red' (This can be adjusted as needed)
    RED_THRESHOLD = 100
    GREEN_THRESHOLD = 45
    BLUE_THRESHOLD = 45

    # Detect whether the central pixel is red enough
    if red > RED_THRESHOLD and green < GREEN_THRESHOLD and blue < BLUE_THRESHOLD:
        # If red enough, it means there's a red obstacle ahead, so turn right
        left_motor.setVelocity(2)  # positive speed turns the wheel forward
        right_motor.setVelocity(-2)  # negative speed turns the wheel backward
    else:
        # If not red enough, it means the path is clear, so move forward
        left_motor.setVelocity(5)
        right_motor.setVelocity(5)
```

EXAMPLES 8

The ROBOT will move forward If the **RED obstacle (ROCK)** the Robot Turn Right, If the Obstacle BLUE the robot will move Backward.

epuck_obstacle_detection_navigation.py

MIDOCEAN UNIVERSITY

EXAMPLES

Webots

robot simulation

Day
02

Color-Based Navigation for a Webots Robot Using Camera Input 3X3 Matrix

MIDOCEAN UNIVERSITY

Analyze a 3x3 matrix around the central pixel and use the average color values for decision-making.

P1	P2	P3
P4	P5	P6
P7	P8	P9

-1	0	+1
-1	0	+1
-1	0	+1

We'll calculate the average red, green, and blue values of the nine pixels at the center of the image and use these averages to determine the robot's movement.

Simulation Loop to Analyze a 3x3 Pixel Matrix


```
while robot.step(TIME_STEP) != -1:
    image = camera.getImage()
    width = camera.getWidth()
    height = camera.getHeight()

    # Calculate the central pixel's coordinates
    centerX = width // 2
    centerY = height // 2

    # Initialize sums for the color components
    sum_red = 0
    sum_green = 0
    sum_blue = 0
```

```
# Loop over a 3×3 matrix around the central pixel
for dx in range(-1, 2):  # -1, 0, 1
    for dy in range(-1, 2):  # -1, 0, 1
        # Calculate the pixel's coordinates
        x = centerX + dx
        y = centerY + dy

        # Sum the color components of the pixel
        sum_red += camera.imageGetRed(image, width, x, y)
        sum_green += camera.imageGetGreen(image, width, x, y)
        sum_blue += camera.imageGetBlue(image, width, x, y)
```

-1	0	+1
-1	0	+1
-1	0	+1

```
# Calculate the average color components
avg_red = sum_red // 9
avg_green = sum_green // 9
avg_blue = sum_blue // 9
```

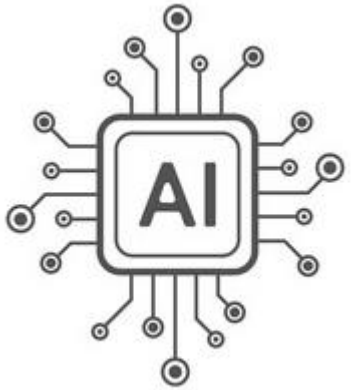
Main Simulation Loop to Analyze a 3x3 Pixel Matrix


```
# Adjust thresholds and decisions based on the average color
    RED_THRESHOLD = 100
    GREEN_THRESHOLD = 45
    BLUE_THRESHOLD = 45

    RED_THRESHOLD2 = 45
    GREEN_THRESHOLD2 = 45
    BLUE_THRESHOLD2 = 100

    if avg_red > RED_THRESHOLD and avg_green < GREEN_THRESHOLD and avg_blue < BLUE_THRESHOLD:
        left_motor.setVelocity(5)  # Turn right
        right_motor.setVelocity(-5)
    elif avg_blue > BLUE_THRESHOLD2 and avg_green < GREEN_THRESHOLD2 and avg_red < RED_THRESHOLD2:
        left_motor.setVelocity(-5)  # Move backward
        right_motor.setVelocity(-5)
    else:
        left_motor.setVelocity(5)  # Move forward
        right_motor.setVelocity(5)
```

Day

02

MIDOCEAN UNIVERSITY



Webots

robot simulation

MIDOCEAN UNIVERSITY

Control the robot's speed based on color detection—moving faster if the color is green and very slowly if the color is red.

We are going to use Machine Learning Algorithm.
We are going to use Linear regression.

Let's assume:
•A high "color score" represents green, encouraging faster movement.
•A low "color score" represents red, dictating slower movement.

For simplicity, let's say our model has determined the relationship between color score and speed as $y=0.2x-1$,
where *y* is the robot's speed,
and *x* is the color score (ranging from 0 for red to 5 for green).

- Color Score: This is a conceptual representation of how "green" or "red" the camera's view is. (by averaging the RGB values).

- **Linear Regression Model:** The model $y = 0.2x - 1$ is a simple formula we're using to decide the robot's speed based on the detected color. The specific coefficients (0.2 and -1) are placeholders for this example and would be determined through actual model training in a real-world application.

- Speed Adjustment: The speed is adjusted based on the color score, with a higher score (more green) resulting in faster speed and a lower score (more red) resulting in slower speed or stopping.

```
from controller import Robot, Camera

# Initialization
robot = Robot()
timestep = int(robot.getBasicTimeStep())

# Assuming motor and camera initialization here
camera = robot.getDevice('camera')
camera.enable(timestep)

left_motor = robot.getDevice('left wheel motor')
right_motor = robot.getDevice('right wheel motor')
left_motor.setPosition(float('inf'))  # Enable speed control
right_motor.setPosition(float('inf'))
left_motor.setVelocity(0)  # Initially stop
right_motor.setVelocity(0)
```

```
while robot.step(timestep) != -1:
    # Simulate getting a color score from the camera
    # Let's say color score calculation logic has been applied here
    # For simplification, assume we get this score directly from the camera
    # A real implementation would involve analyzing the camera image
    color_score = 3  # Placeholder for the actual color score calculation (0 = red, 5 = green)

    # Apply the linear regression model
    # For our example, the model equation is y = 0.2x – 1
# نموذج مبسط للتحكم في السرعة بناءً على درجة اللون
    predicted_speed = 0.2 * color_score - 1  # Simplified model to control speed based on color score

    # Ensure the speed is within the robot's operational range
    predicted_speed = max(-10, min(10, predicted_speed))

    # Set the robot's motors to the predicted speed
    left_motor.setVelocity(predicted_speed)
    right_motor.setVelocity(predicted_speed)
```

- Reading Distance: The robot uses its distance sensor to measure the distance to an object.
- Applying Linear Regression: The linear regression model calculates the speed based on the distance.
- Adjusting Speed: The robot adjusts its speed and direction based on the calculated speed, moving faster when the object is far away and slower as it gets closer.

Linear Regression Model:

Defining the Model Coefficients:

```
a = -0.1  # Coefficient for the distance
b = 5     # Intercept
```

Reading Distance and Calculating Speed:

```
while robot.step(timestep) != -1:
    distance = distance_sensor.getValue()
    speed = a * distance + b
    speed = max(-10, min(10, speed))
    left_motor.setVelocity(speed)
    right_motor.setVelocity(speed)
    print(f"Distance: {distance}, Speed: {speed}")
```

Day
02

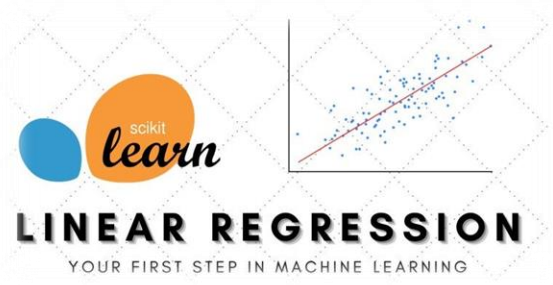
Machine Learning (ML) Algorithm

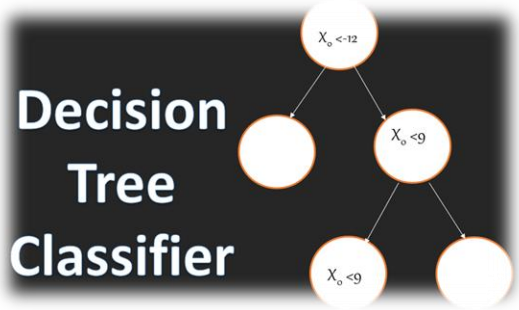
MIDOCEAN UNIVERSITY

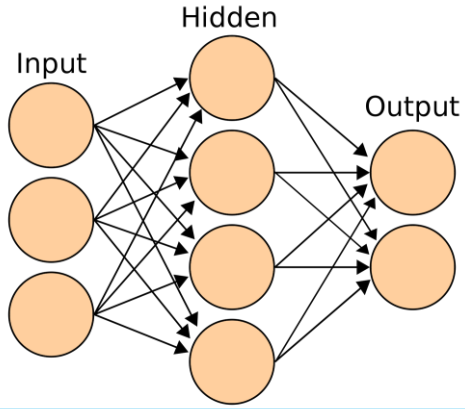
Webots

robot simulation

scikit learn
machine learning in Python







Day

02

MIDOCEAN UNIVERSITY

مجموعة جامعة
الجميع الذكية
للتعليم القابضة

MIDOCEAN
UNIVERSITY

Thank You