Lab 1: Reactive Behaviors and State Machines

CSCI 3302: Introduction to Robotics

Report due 1/27/26 @ 11:59pm

The goals of this lab are to understand

- How to setup a world in the Webots simulation software
- How to retrieve data from the sensors mounted on the e-puck robot
- Position and velocity control of the robot
- How to use state machines to implement a basic controller

You need:

- A functional Python (3.10 to 3.13) development environment
- Webots robot simulation software (https://cyberbotics.com/)

Overview

A very simple way to program a robot is to directly map its sensor input to appropriate wheel motion. Some examples include obstacle avoidance (i.e., "Turn right if you see an obstacle in front of you"), light following/avoidance, and line following. A specific type of robot under this category is a "Braitenberg Vehicle".

Despite the advantages of this approach, it would be quite limiting if robots **always** exhibit the same behavior when presented with stimuli. To achieve more complex behaviors, robots need to switch between different operational modes based on the context they're in.

In the first part of this lab, you get to experiment with different standard behaviors. Then, create a simple finite state machine and switch between different behaviors to accomplish a complex task.

Instructions

Each group must develop their own software implementation and turn in a lab report. **You are encouraged to engage with your lab partners for collaborative problem-solving**, but are

expected to understand and be able to explain everything in your write-up. If your group does not finish the implementation by the end of the class, you may continue this lab on your own time as a complete group.

Part 1: Introduction to Webots

Part 1: Webots Setup + First Simulation (Course version: R2025a)

0) Before you install (2-minute checklist)

- You must use Webots R2025a for this course unless we explicitly say otherwise (Version mismatches can cause annoying differences in worlds/controllers).

- Your computer should support OpenGL 3.3+ (Webots strongly prefers recent NVIDIA/AMD graphics + up-to-date drivers).

- If Webots launches but the graphics look broken / it crashes / you see OpenGL warnings, jump to the Troubleshooting section at the end.

1) Download Webots R2025a (all OS)

- Go to the Webots Download Page and locate R2025a: https://cyberbotics.com/#download

- Download the correct installer for your OS:

- Windows: webots-R2025a_setup.exe
- macOS: .dmg for R2025a (drag-and-drop install is typical)
- Linux: usually a .deb (Ubuntu/Debian) or an archive; use the package that matches your distro/architecture

If you can't find R2025a on the main download page, use GitHub Releases and choose the stable tag for R2025a (avoid nightly builds).

2) Install steps by Operating System

A) Windows 10/11

- Locate the downloaded file: webots-R2025a_setup.exe.

- Double-click the installer → proceed through the prompts (default settings are fine).

- Launch Webots from the Start Menu.

- Verify version: in Webots, go to Help → About and confirm it shows R2025a.

- If Webots opens normally, go to Section 3 (First Simulation + Screenshot).

B) macOS (Intel or Apple Silicon)

- Open the downloaded R2025a .dmg.

- If you cannot open it, and see the security warning, please follow the following step.

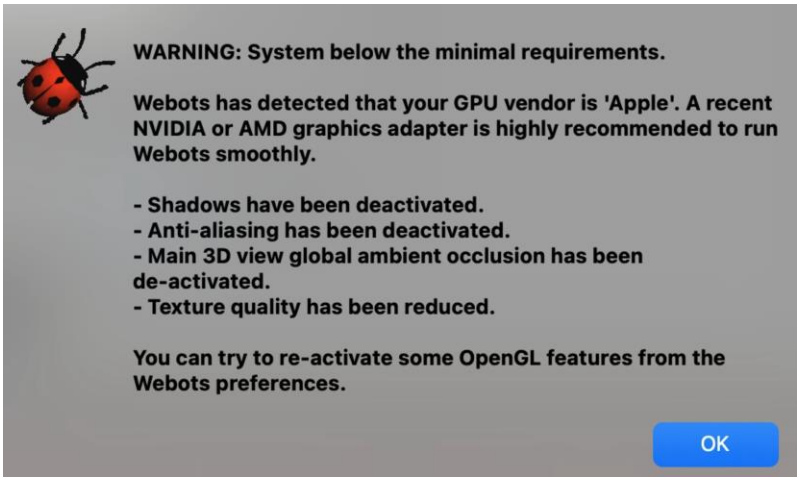
-Go to **Privacy & Security** and scroll down, then you should see a message saying "**webots-R2025a.dmg**" was blocked. Click the **Open Anyway** button, and later you will need to enter your Mac password to confirm it. After opening it, please follow the following instruction to complete your installation.

- Drag Webots into your Applications folder (common macOS install pattern).

- First launch:

- If macOS blocks the app, please allow it in System Settings: go to **Privacy & Security** → scroll down to find a message saying "**webots-R2025a.dmg**" was blocked → Click the **Open Anyway** button → Enter your Mac password or Fingerprint to confirm it.

- If Webots opens normally, go to Section 3 (First Simulation + Screenshot). If you see the warning of the minimal requirement, ignore it and click OK. But in the future, if we need these features, we can navigate to Webots > Preference in the top menu bar and manually set these features.



C) Linux (recommended: Ubuntu/Debian for easiest install)

- Option 1 (recommended for class consistency): Install from a .deb

- Download the R2025a .deb package.

- Install it using either:

- GUI: double-click the .deb and install via your Software Center, or

- Terminal (common approach):

- sudo apt install ./webots-<R2025a_package_name>.deb

- Launch Webots (from Applications menu or by running webots in a terminal).

- Verify version: Help → About → R2025a

- If Webots opens normally, go to Section 3 (First Simulation + Screenshot).

3) Required: Tutorial 1 – First Simulation.

https://cyberbotics.com/doc/guide/tutorials

You need to follow Tutorial 1: Your First Simulation. It walks you through creating a project directory, adding an arena/world elements, and running a first simulation. **When running simulation, please use python instead of C or other languages.**

After completing it, please follow the instruction to take a screenshot, and save the world file (.wbt), and Controller file (.py). Please follow the tutorial to name the world file and controller file.

Please note that: if you see the **WARNING: Python was not found,** you can follow the instruction to fix it.

4) Required: Tutorial 4

Optional: Tutorial 2/3

https://cyberbotics.com/doc/guide/tutorials

Do these in order

- Tutorial 1: First Simulation (required)

- Tutorial 2: Modifying the Environment (optional)

Updated 9/9/2020 to add Hints/Clarifications/Extended Due Date

- Tutorial 3: Modifying Appearance (optional)

- Tutorial 4: Creating a Robot Controller (required)

After completing tutorials, please follow the instruction to take a screenshot, and save the world file (.wbt), and Controller file (.py). Please follow the tutorial to name the world file and controller file.

Troubleshooting (common issues)

- "Webots requires OpenGL …" / blank 3D view / crashes on start
  Solution: Webots requires OpenGL 3.3+ and recommends recent NVIDIA/AMD GPUs with updated drivers. Follow the official steps to verify your graphics driver and update it as needed.
- Webots is slow / stuttery
  Solution: This is often a graphics driver issue; updating GPU drivers is the first fix.
- Can't find R2025a on the main site
  Solution: Use the official GitHub Releases page and select the stable R2025a release (avoid nightly builds).

- when working on **Webots Tutorial 1**, you may see the **WARNING: Python was not found**
  - If you do, you can fix it by following the instruction to fix:
    - Open Terminal and run: which python3
    - Copy the full path it returns (e.g., /usr/local/bin/python3 or /opt/homebrew/bin/python3)
    - Go to Webots → Preferences → General → Python command
    - Paste the full path you copied into the Python command field
    - Click OK, then click the Reload button ( ) in the simulation toolbar and run the simulation.

Part 2: World Creation and Controller Programming

Open lab1.wbt world in webots.

Implement a state machine-based controller that recreates the following behavior:

Updated 9/9/2020 to add Hints/Clarifications/Extended Due Date

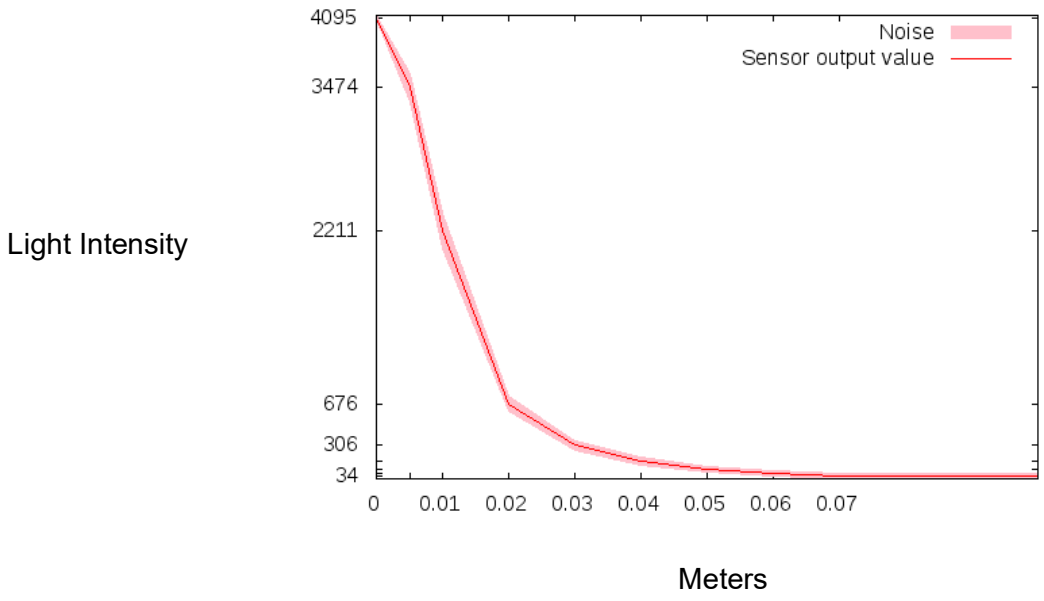
The robot follows the wall on its left side, adjusting to maintain a consistent distance from the wall while moving straight. If the robot encounters an obstacle in front, the robot makes a right turn to avoid the obstacle before continuing to follow the wall. Once the robot detects the light source from left, front, and right light sensors (ls0, ls1, ls2, ls5, ls6, ls7) it will make a 180 degree turn and follow the wall on its right side. When the robot detects the light source again, it stops.

HINT: Planning your answer for Part 3 - #3 before writing code will make this easier.

HINT: You can manually hardcode how long the robot should actuate its wheels to turn around, or you can use the distance sensors behind it to tell when you've finished turning.

HINT: The distance sensor doesn't return a distance value; it returns an intensity measurement: you will need to convert the returned value to a distance, which will only be approximate given the low accuracy of this sensor.

You can use the lookup table below to map distance sensor values to distances:



HINT: The distance sensor on the e-Puck is very noisy! You can right-click the robot and click "Show Robot Window" to view sensor readings. You can then place the robot near obstacles and see what the sensor values are to use as thresholds in your code.

Updated 9/9/2020 to add Hints/Clarifications/Extended Due Date

Part 3: Lab Report

Create a report that includes/answers the following:

1. The names of everyone in your group

2. Screenshots of your world from Tutorial 1 and 4. In your screenshot, it should have:

- Your arena/world visible in the 3D view, and
- The Scene Tree (left panel) visible, and
- The project name/folder visible somewhere (title bar or file tree).
- Suggested naming (to keep grading easy)

3. A drawing of your state machine. Make sure all the states and transitions are labeled and that it is faithful to your implementation.

4. Were you able to get your controller to complete the task? If not, which parts failed? Why?

5. A statement indicating whether you have worked with Python before, and if so, describe your experience.

6. How much time did you spend programming **Part 2** of this lab?

Please submit a zip file containing your Lab Report in PDF form, your Webots world file (.wbt), and Controller file (.py) on Canvas. We only need one submission per group.

Updated 9/9/2020 to add Hints/Clarifications/Extended Due Date