Robo Desk Buddy - Sprint Documentation

Week 1 - Robot Dev

Sprint 1 (Setup + Repo + Project Scaffolding)

Deliverables:

- ✅ GitHub repo created for robot code
- ✅ Webots installed and verified on dev machine
- ✅ Minimal world file created with robot loaded
- ✅ Basic controller created that executes placeholder action
- ✅ Documentation of setup steps

Unit Tests:

- World file opens in Webots without errors
- Controller runs and executes placeholder action successfully

To-Dos:

- ☑ Initialize GitHub repo
- ☑ Install Webots
- ☑ Create minimal world file
- ☑ Add basic controller
- ☑ Document setup process

Prerequisites & Installation

Required Software

1. **Webots Robot Simulator**

- Download from: https://cyberbotics.com/
- Version: R2023b or later
- Install following platform-specific instructions

2. **Python 3.8+**
   - Webots includes Python, but you can configure external Python if needed
3. **Git**
   - For version control and collaboration
4. **Code Editor** (Optional but recommended)
   - Visual Studio Code with Python extension

Setup Instructions

Step 1: Create Project Directory

```
cd Desktop
mkdir robo_desk_buddy
cd robo_desk_buddy
```

Step 2: Initialize Webots Project

1. Open Webots
2. Go to: **Wizards → New Project Directory…**
3. **Locate and select** the `robo_desk_buddy` folder you created
4. Give the world a name (e.g., `robo_world`)
5. Click **Finish**

This creates the following structure:

```
robo_desk_buddy/
├── controllers/
├── worlds/
└── protos/ (if needed)
```

Step 3: Exit External Projects (if needed)

- If you had any other Webots projects open, close them
- Work only in your new `robo_desk_buddy` project

Step 4: Create the World File

Add Robot to World

1. In Webots, open your world file from `worlds/robo_world.wbt`
2. Click the **Add** button (or right-click scene tree → Add New)
3. Select a robot base (e.g., `Robot` or duplicate an existing robot child class)
4. Rename the robot to `robo_desk_buddy`

Add Environment Objects

Add these objects to create the desk environment:

- **Table/Desk**: Add a `Solid` object with a box shape
- **Wall**: Add another `Solid` with appropriate dimensions
- **Floor**: Use the default floor or add a custom one

Step 5: Configure Viewpoint

The viewpoint controls how the camera is positioned in the simulation.

Method 1: Mouse Controls (Easier)

- **Zoom in**: Scroll mouse wheel forward
- **Zoom out**: Scroll mouse wheel backward
- **Tilt up**: Click and drag upward
- **Tilt down**: Click and drag downward
- **Rotate left**: Click and drag left
- **Rotate right**: Click and drag right

When you're satisfied with the view:

1. Press **Ctrl + Shift + R** to reload
2. Click **Save** when prompted

Method 2: Manual Edit (VS Code or Webots)

1. Look at the left pane in Webots
2. Find the `Viewpoint` node
3. Adjust these fields:
   - `orientation` : Direction the camera faces
   - `position` : X, Y, Z coordinates of camera
   - `fieldOfView` : Zoom level

You can edit these values directly in:

- Webots Scene Tree (left pane)
- OR in VS Code by opening the `.wbt` file

Step 6: Create the Controller

Create Controller Directory and Files

```
cd controllers
mkdir robo_desk_buddy
cd robo_desk_buddy
touch robo_desk_buddy.py
```

Your structure should now look like:

```
robo_desk_buddy/
├── controllers/
│   └── robo_desk_buddy/
│       └── robo_desk_buddy.py
└── worlds/
    └── robo_world.wbt
```

Basic Controller Template

Create `robo_desk_buddy.py` with this basic structure:

```
from controller import Robot
```

```
# Create robot instance
robot = Robot()

# Get simulation timestep
timestep = int(robot.getBasicTimeStep())

# Main control loop
while robot.step(timestep) != -1:
    # Placeholder action will go here
    pass
```

Step 7: Link Controller to Robot

1. In Webots, select your `robo_desk_buddy` robot in the scene tree
2. In the properties panel, find the `controller` field
3. Set it to `robo_desk_buddy` (the name of your controller folder)

Step 8: Add Devices to Robot (Optional for Sprint 1)

Adding Devices in World File

1. Select your robot in the scene tree
2. Expand the robot node
3. Right-click on robot → **Add New** → Choose device type (LED, Motor, Sensor, etc.)
4. Name each device appropriately

Accessing Devices in Controller

```
from controller import Robot

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

# Get devices by name
# Example: led = robot.getDevice("led_name")

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

```
    # Control devices here
    pass
```

Note: For Sprint 1, you can experiment with simple placeholder devices. Any device you want to use must first be added to the robot in the world file, then accessed via `robot.getDevice("device_name")` in the controller.

GitHub Setup

Initialize Git Repository

```
cd robo_desk_buddy
git init
git add .
git commit -m "Initial commit: Webots Robo Desk Buddy setup"
```

Create GitHub Repository

1. Go to GitHub → **New Repository**
2. Name: `robo-desk-buddy`
3. Don't add README, .gitignore, or license (we'll create them)
4. Copy the repository URL

Connect Local to GitHub

```
git remote add origin <your-repo-url>
git branch -M main
git push -u origin main
```

Clone for Teammates

```
git clone <your-repo-url>
cd robo-desk-buddy
# Open the project in Webots
# Run simulation
```

Collaboration Workflow

```
 # Before starting work
git pull origin main

# After making changes
git add .
git commit -m "Descriptive message"
git push origin main
```

Running the Simulation

1. Open Webots
2. Open `worlds/robo_world.wbt`
3. Click the **Play** button (▶️) to start simulation
4. The controller should run without errors
5. Check the console for any output/logs

Verification Checklist

Sprint 1 Complete When:

- [ ] World file opens in Webots without errors
- [ ] Robot appears in the scene with table and wall
- [ ] Viewpoint is properly configured
- [ ] Controller file exists and is linked to robot
- [ ] Simulation runs without crashes
- [ ] GitHub repo is created and code is pushed
- [ ] This documentation exists in README.md

Project Structure

```
robo_desk_buddy/
├── controllers/
│   └── robo_desk_buddy/
│       └── robo_desk_buddy.py
├── worlds/
│   └── robo_world.wbt
├── README.md
└── .gitignore (recommended)
```
