
Sprint 2 (Placeholder Robot Action - Manual Trigger)

Deliverables:

- ✅ Controller updated to include at least one placeholder behavior (e.g., dance, wave, or move forward)
- ✅ Action can be triggered manually in simulation using keyboard
- ✅ Controller refactored into a proper Robot subclass
- ✅ Keyboard device integrated for manual control
- ✅ Documentation of how action was implemented and tested

Unit Tests:

- Placeholder action executes without errors
- Logs confirm successful execution
- Keyboard input triggers correct action

To-Dos:

- [x] Refactor controller into Robot subclass
- [x] Add Keyboard device to world file
- [x] Implement keyboard input handling
- [x] Add placeholder action to controller
- [x] Run simulation and trigger action
- [x] Verify logs
- [x] Document implementation steps

Sprint 2 Setup Instructions

Step 1: Refactor Controller to Use Class Structure

Transform your basic controller into a proper Robot subclass for better organization and extensibility.

Update `robo_desk_buddy.py`:

```
from controller import Robot

class RoboDeskBuddy(Robot):
    def __init__(self):
        # Initialize parent Robot class
        super().__init__()

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

        # Get devices (add your devices here)
        # Example: self.led = self.getDevice("led_name")

    def placeholder_action(self):
        """Define your placeholder action here"""
        print("Executing placeholder action...")
        # Add your action code here

    def run(self):
        """Main control loop"""
        while self.step(self.timestep) != -1:
            # Your control logic here
            pass

# Create robot instance and run
if __name__ == "__main__":
    robot = RoboDeskBuddy()
    robot.run()
```

Key Changes:

- Created `RoboDeskBuddy` class that inherits from `Robot`
- `__init__` method calls `super().__init__()` to inherit Robot properties
- Device initialization in `__init__`
- `self` keyword used throughout to access class properties and methods
- Organized structure for adding multiple functions

Step 2: Add Keyboard Device to World File

Add External Proto at Top of World File

Open `worlds/robo_world.wbt` in VS Code or text editor and add this line at the **very top** of the file:

```
EXTERNPROTO "https://raw.githubusercontent.com/cyberbotics/webots/R2
```

Add Keyboard as Child of Robot

1. Open world file in Webots
2. Select your `robo_desk_buddy` robot in scene tree
3. Expand the robot node
4. Right-click on robot → **Add New** → **Keyboard**
5. The Keyboard device should now appear as a child of your robot

Step 3: Enable Keyboard in Controller

Update Controller with Keyboard Support:

```
from controller import Robot, Keyboard

class RoboDeskBuddy(Robot):
    def __init__(self):
        super().__init__()
        self.timestep = int(self.getBasicTimeStep())

        # Initialize keyboard
        self.keyboard = self.getKeyboard()
        self.keyboard.enable(self.timestep)

        # Get other devices
        # self.led = self.getDevice("led_name")
        # self.motor = self.getDevice("motor_name")

    def blink_lights(self):
```

```
        """Placeholder action: Blink LED lights"""
        print("💡 Blinking lights!")
        # Add LED blinking code here

    def wave(self):
        """Placeholder action: Wave gesture"""
        print("👋 Waving!")
        # Add wave motion code here

    def run(self):
        """Main control loop with keyboard input"""
        print("Robot ready! Press keys to trigger actions:")
        print("  B - Blink lights")
        print("  W - Wave")

        while self.step(self.timestep) != -1:
            # Get keyboard input
            key = self.keyboard.getKey()

            # Check if a key was pressed
            if key != -1:
                # Use ord() to convert character to key code
                if key == ord('B'):
                    self.blink_lights()
                elif key == ord('W'):
                    self.wave()

# Create robot instance and run
if __name__ == "__main__":
    robot = RoboDeskBuddy()
    robot.run()
```

Key Components:

1. **Import Keyboard**: `from controller import Robot, Keyboard`

2. **Get Keyboard Device**:

```
self.keyboard = self.getKeyboard()
```

3. **Enable Keyboard**:

```
self.keyboard.enable(self.timestep)
```

4. **Read Key Input**:

```
key = self.keyboard.getKey()
```

5. **Check Key Pressed**:

```
if key != -1:  # -1 means no key pressed
```

6. **Use ord() for Key Codes**:

```
if key == ord('B'):  # Check for 'B' key
```

Step 4: Run and Test

1. Save your controller file
2. Open world file in Webots
3. Click **Play** (▶️)
4. Click inside the 3D view to focus (important for keyboard input!)
5. Press **B** to trigger blink lights
6. Press **W** to trigger wave
7. Check console for log messages

Sprint 3 Preview (Expanded Robot Action Demo)

Deliverables:

- Controller extended with additional placeholder action (e.g., turn + speak)
- Robot can switch between actions during simulation
- Documentation of how multiple actions were wired together

Unit Tests:

- Robot executes second action without errors
- Robot can switch between first and second action during a single run

To-Dos:

- [ ] Add second placeholder action
- [ ] Update controller to handle multiple actions
- [ ] Run and verify both actions in sequence
- [ ] Document process

Troubleshooting

Sprint 2 Specific Issues

Keyboard not responding

- Make sure to **click inside the 3D view** to focus the simulation
- Verify Keyboard device is added as child of robot in world file
- Check that EXTERNPROTO line is at the top of .wbt file
- Ensure `keyboard.enable(self.timestep)` is called in `__init__`

Key codes not working

- Use `ord('KEY')` for letter keys (e.g., `ord('B')`, `ord('W')`)
- Keys are case-sensitive
- Check console output to see what key code is being received: `print(f"Key pressed: {key}")`

Actions not executing

- Add print statements to verify function is being called
- Check indentation in if/elif statements
- Verify `key != -1` check is present

Sprint 1 Issues

World file won’t open

- Check that all file paths are correct
- Ensure Webots version is compatible
- Look for syntax errors in .wbt file

Controller not running

- Verify controller name matches folder name
- Check that controller field in robot is set correctly
- Look for Python syntax errors in console

GitHub push fails

- Ensure you have internet connection
- Verify remote URL is correct: `git remote -v`
- Check that you have push permissions to the repo

Notes

- All changes to the world file should be saved before running
- Use **Ctrl + Shift + R** to reload the world
- Check Webots console for error messages and logs
- Document any device additions for future sprints

Sprint 1 Status: ✅ Complete
Sprint 2 Status: ✅ Complete
Next Sprint: Sprint 3 - Expanded Robot Action Demo

Example Code Snippets

Complete Sprint 2 Controller Example

```
from controller import Robot, Keyboard
```

```
class RoboDeskBuddy(Robot):
    def __init__(self):
        """Initialize robot and devices"""
        super().__init__()
        self.timestep = int(self.getBasicTimeStep())

        # Initialize keyboard
        self.keyboard = self.getKeyboard()
        self.keyboard.enable(self.timestep)

        print("🤖 Robo Desk Buddy initialized!")

    def blink_lights(self):
        """Action 1: Blink LED lights"""
        print("💡 Blinking lights!")
        # TODO: Add LED control code
        # Example: self.led.set(1)

    def wave(self):
        """Action 2: Wave gesture"""
        print("👋 Waving!")
        # TODO: Add motor control for waving
        # Example: self.arm_motor.setPosition(1.5)

    def run(self):
        """Main control loop"""
        print("=" * 50)
        print("🎮 Controls:")
        print("  B - Blink lights")
        print("  W - Wave")
        print("=" * 50)

        while self.step(self.timestep) != -1:
            # Get keyboard input
            key = self.keyboard.getKey()

            # Process keyboard input
            if key != -1:
```

```
                if key == ord('B'):
                    self.blink_lights()
                elif key == ord('W'):
                    self.wave()
                else:
                    print(f"⚠️ Unknown key: {chr(key)}")

# Entry point
if __name__ == "__main__":
    robot = RoboDeskBuddy()
    robot.run()
```

Adding More Actions (Sprint 3 Preview)

```
def dance(self):
    """Action 3: Dance move"""
    print("💃 Dancing!")
    # TODO: Add coordinated motor movements

def speak(self):
    """Action 4: Speak/beep"""
    print("🔊 Speaking!")
    # TODO: Add speaker control

# In run() method, add:
elif key == ord('D'):
    self.dance()
elif key == ord('S'):
    self.speak()
```

Tips for Debugging

1. **Add print statements** everywhere to track execution
2. **Check console output** for error messages
3. **Use descriptive log messages** with emojis for clarity
4. **Test one action at a time** before adding more

5. **Verify device names** match between world file and controller
6. **Use ChatGPT/Claude** when stuck on errors

Sprint 1 Status: ✅ Complete
Sprint 2 Status: ✅ Complete
Next Sprint: Sprint 3 - Expanded Robot Action Demo