AutoPipette Machine Control Panel Documentation

Overview

This module provides a user interface and server-side logic for controlling an automated pipetting system. The UI is built using the Shiny framework, and it integrates various functions to move the robot to specific coordinates, home motors, and perform kit manufacturing and sample preparation tasks.

Table of Contents

Dependencies

Ensure you have the following Python packages installed:

```
pip install shiny asyncio
```

UI Structure

The user interface is built using the Shiny library and is composed of various input controls such as numeric inputs, action buttons, and text inputs. The layout includes a header, sidebar for controls, and a main area for displaying output.

Key UI Components:

- **Numeric Inputs:** To input X, Y, Z coordinates and speed.
- **Action Buttons:** To trigger actions like moving to coordinates, homing motors, and initiating tasks.
- **Text Input:** To specify the name of the location for movement.
- **Output Text:** To display the status and results of operations.

Functions

`move_to_coordinates(x, y, z, speed)`

Moves the robotic arm to specified coordinates with a given speed.

- **Parameters:**
  - `x` (int): The X coordinate.
  - `y` (int): The Y coordinate.
  - `z` (int): The Z coordinate.
  - `speed` (int): Speed of movement.

`get_coordinates_from_name(name)`

Retrieves the coordinate object corresponding to a named location.

- **Parameters:**
  - `name` (str): The name of the location.
- **Returns:**
  - A `Coordinate` object with the corresponding X, Y, Z values.
-

Server-Side Logic

The server logic manages reactive inputs and triggers corresponding actions based on user interaction.

Handlers

Move Handler

Triggered when the "Move to Coordinates" button is pressed. It moves the robotic arm to the specified X, Y, Z coordinates with the defined speed.

HomeX Handler

Handles homing of the X-axis stepper motor.

Home Handler

Moves the robotic arm to the predefined home coordinates.

Kit Handler

Initiates the kit manufacturing process by calling the appropriate function.

Sample Handler

Starts the sample preparation process.

Stop Handler

Stops all ongoing operations.

Move to Location Handler

Moves the robotic arm to a location based on the name provided in the text input.

Traverse Wells Handler

Initiates a well traversal routine, typically involving a pipetting sequence across multiple wells.

Initialize Pipette Handler

Homes all pipette motors, ensuring they are in the correct starting position.

Usage

To run the application, use the following command:

```
python app.py
```

This will start the Shiny server, and you can access the control panel in your web browser by navigating to http://localhost:8000.

Recommended: Using ANGRY IP Scanner to find Manta IP and localhost IP

Additional Code Modules

PipetteController Class

The `PipetteController` class provides methods to control a pipette using servo and stepper motors. It includes functionalities to eject tips, set servo angles, home the stepper motor, and move the stepper motor by a specific distance.

Class Definition:

```
class PipetteController:
    def __init__(self, servo_name, stepper_name):
        self.servo_name = servo_name
        self.stepper_name = stepper_name

    def eject_tip(self):
        self.set_servo_angle(SERVO_ANGLE_RETRACT)
        time.sleep(EJECT_WAIT_TIME)
        self.set_servo_angle(SERVO_ANGLE_READY)
        time.sleep(MOVEMENT_WAIT_TIME)
        self.set_servo_angle(SERVO_ANGLE_RETRACT)

    def set_servo_angle(self, angle):
        gcode_command = f"SET_SERVO SERVO={self.servo_name}
ANGLE={angle}"
        send_gcode(gcode_command)

    def home_stepper(self):
        gcode_command = f"MANUAL_STEPPER STEPPER={self.stepper_name}
SPEED=5 MOVE=-10 STOP_ON_ENDSTOP=1"
        send_gcode(gcode_command)
        gcode_command = f"MANUAL_STEPPER STEPPER={self.stepper_name}
SET_POSITION=0"
        send_gcode(gcode_command)
```

```
    def move_stepper(self, distance, speed=20):
        gcode_command = f"MANUAL_STEPPER STEPPER={self.stepper_name}
SPEED={speed} MOVE={distance}"
        send_gcode(gcode_command)
```

Methods:

- **`__init__(self, servo_name, stepper_name)`**
  - Initializes the pipette controller with the specified servo and stepper motor names.
- **`eject_tip(self)`**
  - Controls the servo to eject the pipette tip.
- **`set_servo_angle(self, angle)`**
  - Sets the servo motor to a specific angle using G-code.
- **`home_stepper(self)`**
  - Homes the stepper motor to its starting position.
- **`move_stepper(self, distance, speed=20)`**
  - Moves the stepper motor by a specified distance at the given speed.

Movement Module

The `movement.py` module contains functions to control the movement of the robotic arm to specific coordinates. It includes functions to move the robot to a given coordinate, home specific axes, and initiate kit manufacturing or sample preparation tasks.

Volume Module

The `volumes.py` module contains definitions and constants related to the liquid volumes used in pipetting operations. This ensures consistency in handling various liquids across different operations.

Coordinates Module

The `coordinates.py` module defines the Coordinate class, which represents 3D coordinates (X, Y, Z) with an optional speed parameter for movement. It includes predefined coordinates for various locations such as 100µL tips, wells, vials, and garbage positions.
