BotEditorPy User Guide

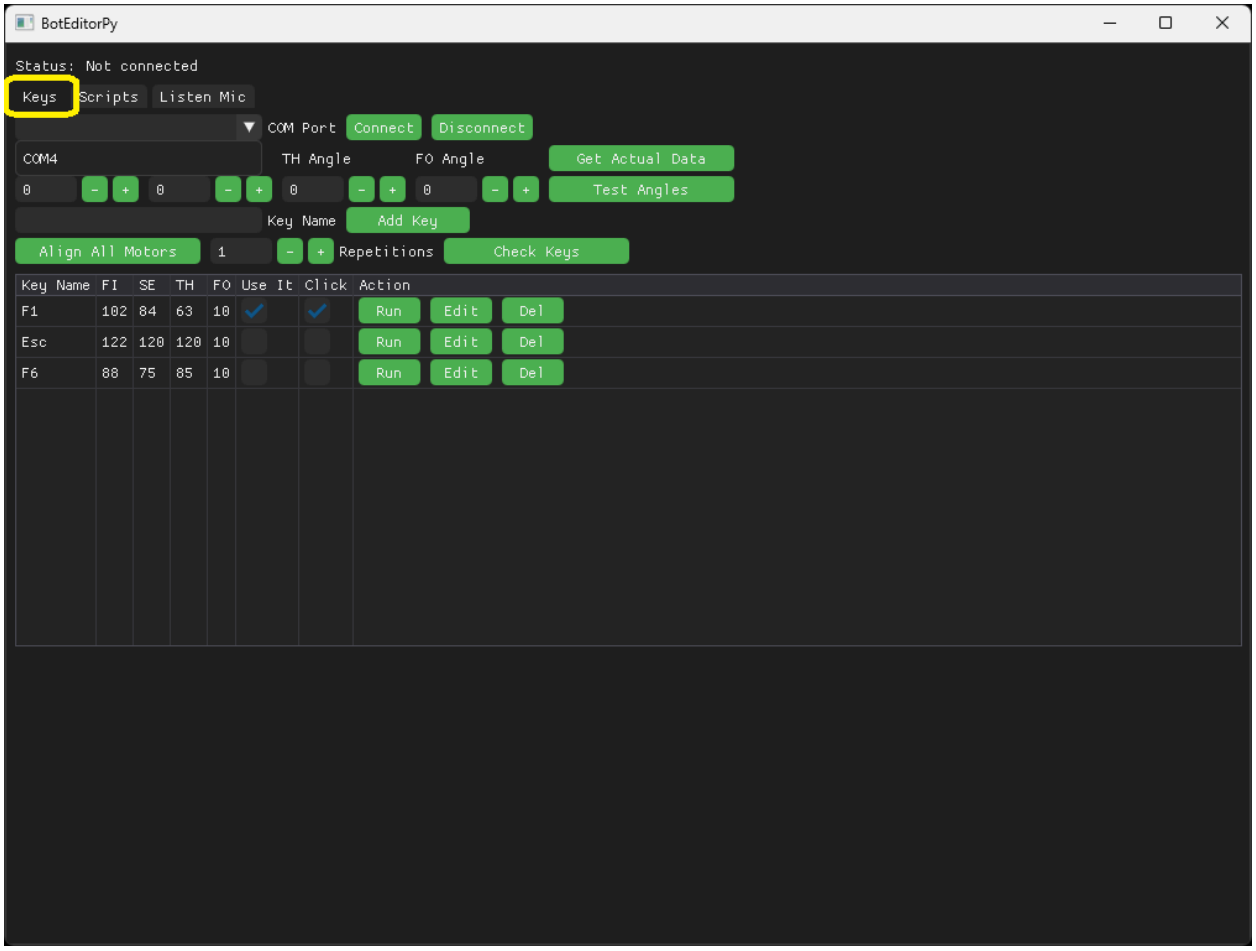
This user guide provides detailed instructions on how to use the `BotEditorPy` application to control a robotic arm via a graphical interface and voice commands. The application is divided into three main tabs: **Keys**, **Scripts**, and **Listen Mic**. This guide will walk you through the functionality of each tab, how to perform common tasks, and tips for troubleshooting.

Table of Contents

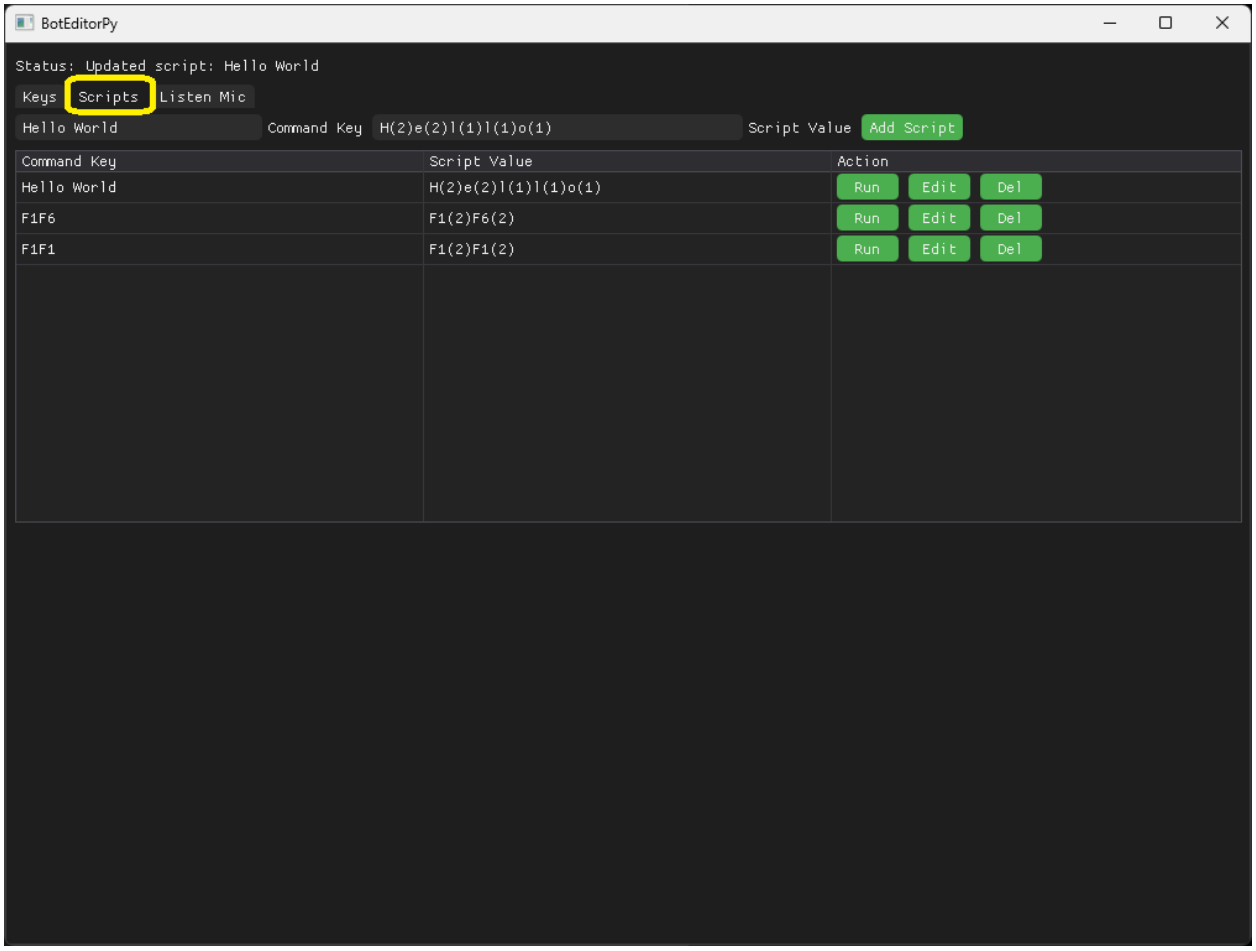
Overview

`BotEditorPy` is a Python application designed to control a robotic arm using motor angles and scripts. It communicates with an Arduino over a serial connection to move the arm and supports voice commands to execute predefined scripts. The application features:

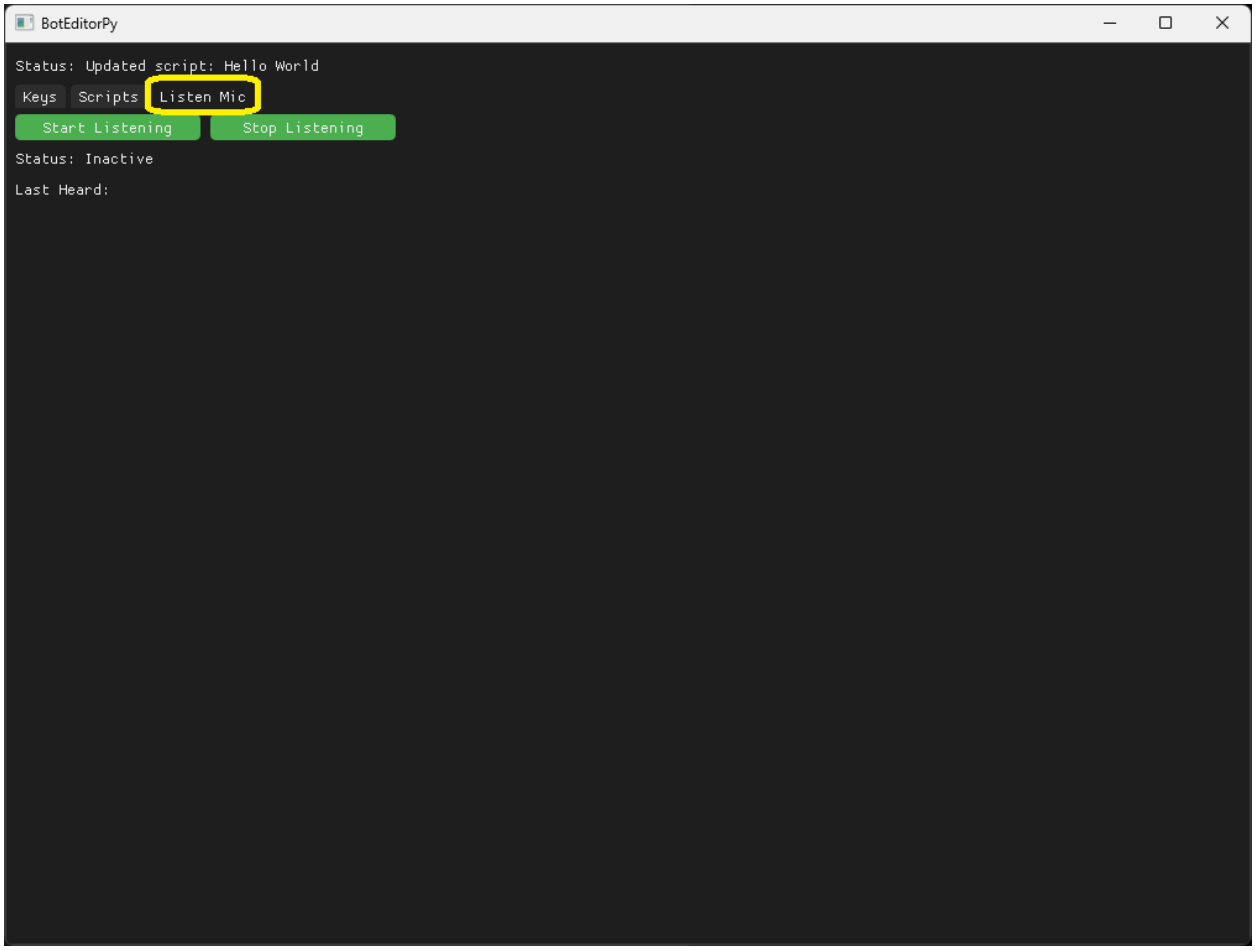
- **Keys Tab**: Manage motor angles for individual keys (e.g., F1, Esc) and save them to `keys.json`.



- **Scripts Tab**: Create and run scripts that sequence key presses with delays (saved in `scripts.json`).

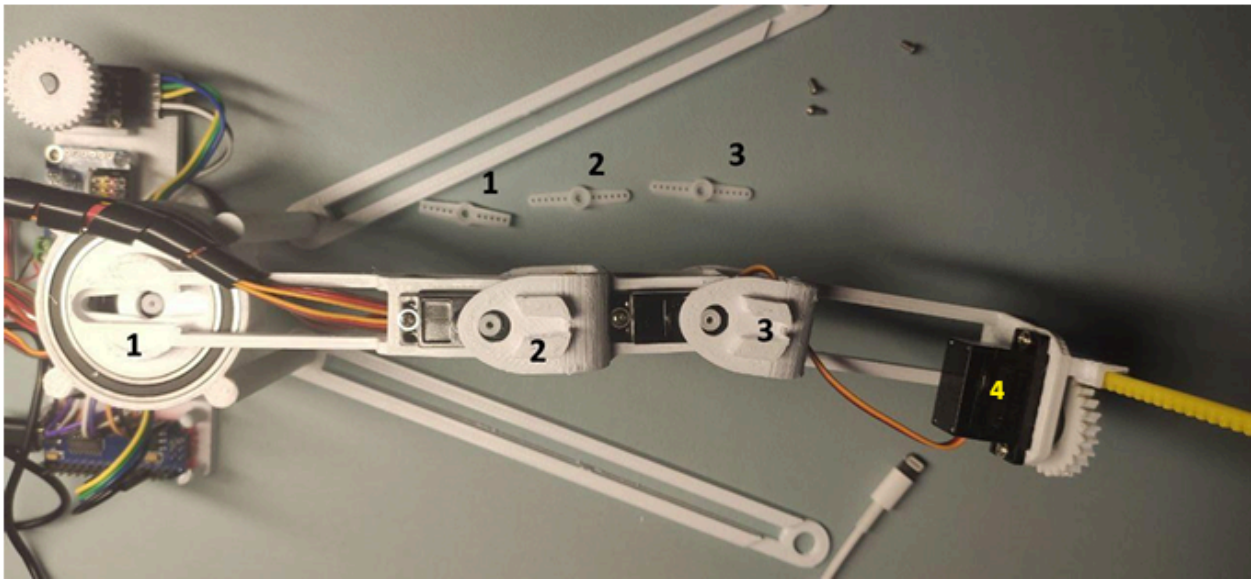


- **Listen Mic Tab**: Use voice commands to execute scripts by speaking their names.

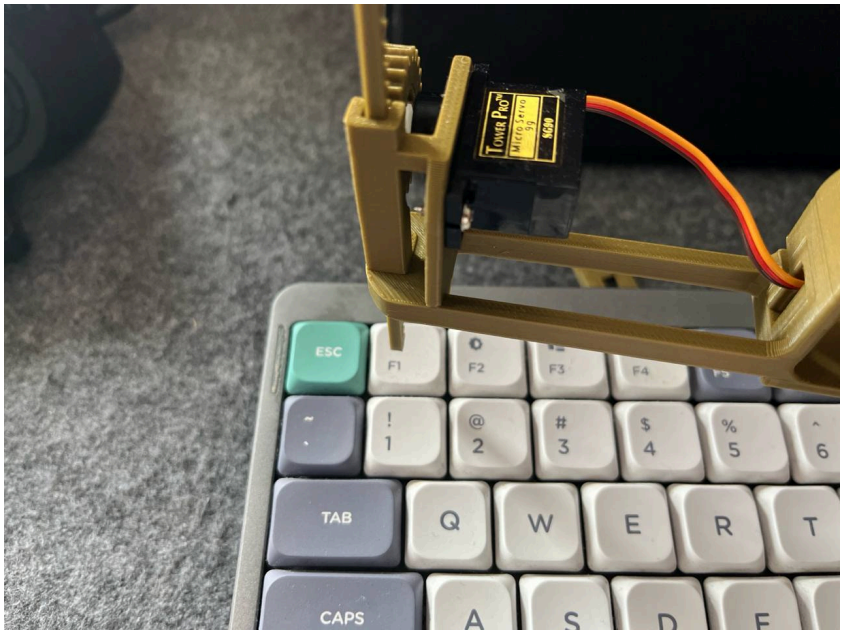


The robotic arm uses four motors:

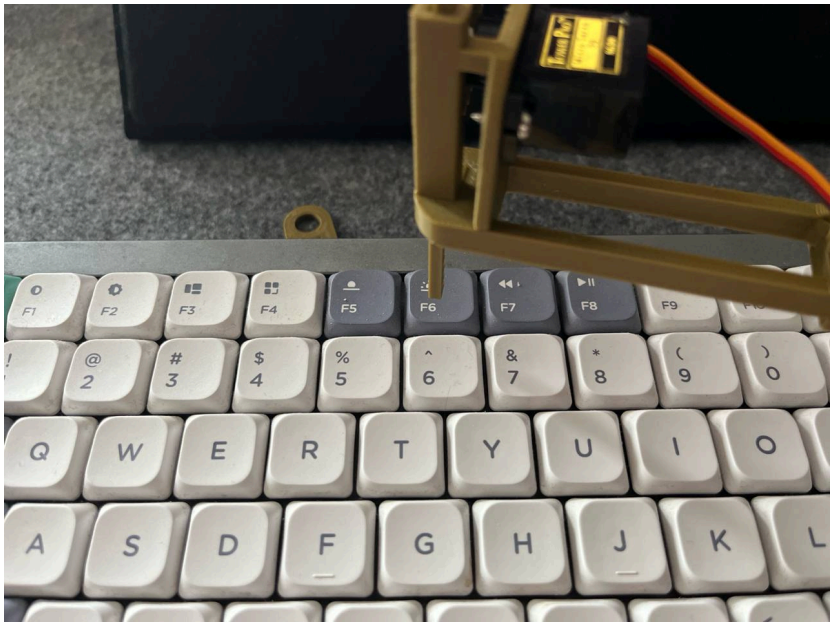
- **FI**: First motor angle.
- **SE**: Second motor angle.
- **TH**: Third motor angle.
- **FO**: Fourth motor angle (used for clicking).



Scripts define sequences of key presses, such as `"F1(2)F6(2)"`, which means pressing the F1 key (with a 2-second delay)



followed by the F6 key (with a 2-second delay).



Getting Started

Before using the application, ensure you have installed it following the instructions in `INSTALL.md`. Once installed:

1. **Launch the Application**:

    - Navigate to the project directory.
    - Activate the virtual environment and run the application:
        - **Windows**:

            cd [YourPathTo]\BotEditorPy

            myenv\Scripts\activate

            python main.py

        - **Linux/macOS**:

            cd /path/to/BotEditorPy

            source myenv/bin/activate

            python main.py

2. **Interface Overview**:

    - The application window opens with three tabs: **Keys**, **Scripts**, and **Listen Mic**.
    - A status bar at the top displays messages (e.g., "Connected to COM4").

3. **Prepare the Arduino**:

    - Ensure your Arduino is connected via USB and running a sketch that can interpret commands in the format `FI:90,SE:45,TH:10,FO:180`.

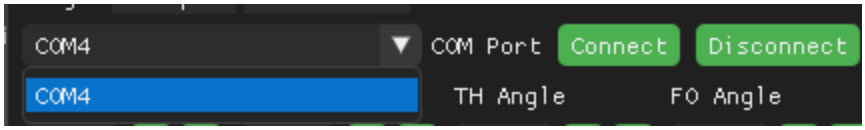
Keys Tab

The **Keys** tab allows you to define motor angles for individual keys, save them, and test them on the robotic arm.

Connecting to the Arduino

1. **Select a COM Port**:

    - At the top of the tab, use the "COM Port" dropdown to select the port where your Arduino is connected (e.g., `COM4` on Windows, `/dev/ttyUSB0` on Linux).



    - If no ports are listed, ensure your Arduino is connected and the drivers are installed.

2. **Connect**:

    - Click the **Connect** button.
    - The status bar should update to "Connected to " (e.g., "Connected to COM3").
    - If it fails, check the Arduino connection and try a different port.

3. **Disconnect** (Optional):

    - To disconnect, click the **Disconnect** button.
    - The status bar will show "Disconnected".

Setting Motor Angles

1. **Adjust Angles**:

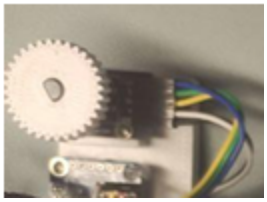
    - Below the COM port section, you'll see input fields for four motors: **FI**, **SE**, **TH**, and **FO**.
    - Enter values between 0 and 200 for each motor (e.g., FI: 102, SE: 84, TH: 63, FO: 10).

2. **Test Angles**:

    - Click the **Test Angles** button to send the angles to the Arduino.
    - The robotic arm should move to the specified angles, and the status bar will show "Sent: FI:102,SE:84,TH:63,FO:10".
    - If the arm doesn't move, ensure the Arduino is connected and the sketch is running.

3. **Get Actual Data**:

- Click the **Get Actual Data** button to retrieve the current motor angles from the Arduino.(For more precise and real-time control, you can use a **rotary encoder** connected to your Arduino)



- The input fields will update with the current angles, and the status bar will show "Angles updated".

Adding or Editing a Key

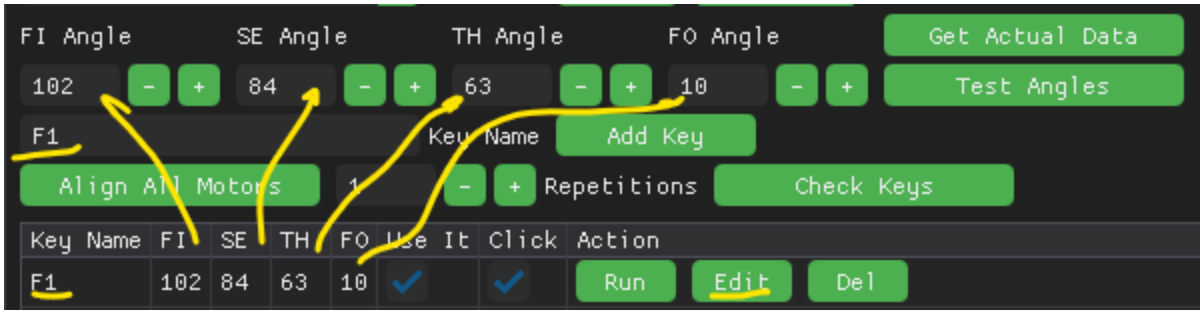
1. **Set Motor Angles**:
   - Adjust the motor angles as described above.
     (For more precise and real-time control, you can use a **rotary encoder** connected to your Arduino)
2. **Enter a Key Name**:

   - In the "Key Name" field, enter a name for the key (e.g., `F1`).

3. **Add or Update the Key**:

   - Click the **Add Key** button.
   - If the key name doesn't exist, it will be added to `keys.json`. If it already exists, it will be updated.
   - The status bar will show "Added key: F1" or "Updated key: F1".
   - The table below will update to show the new or updated key.

4. **Edit an Existing Key**:



- In the table, find the key you want to edit (e.g., F1).
- Click the **Edit** button in the "Action" column.
- The motor angles and key name will load into the input fields, and the status bar will show "Editing F1".
- Adjust the angles or name as needed, then click **Add Key** to save the changes.

Running a Key

1. **Select a Key**:

    - In the table, find the key you want to run (e.g., F1).

2. **Enable Use It and Click (Optional)**:

    - Check the **Use It** checkbox to enable the key for use in sequences (used in "Check Keys").
    - Check the **Click** checkbox to enable a clicking action (extends the FO motor to 190 and back to the original FO value).
    - Note: **Click** can only be enabled if **Use It** is checked.

3. **Run the Key**:

    - Click the **Run** button in the "Action" column.
    - The robotic arm will move to the specified angles (FI, SE, TH).
    - If **Click** is enabled, the FO motor will extend to 190, then return to its original value.
    - The status bar will show "Sent: FI:102,SE:84,TH:63" (and additional FO commands if clicking).

Checking Multiple Keys

1. **Enable Keys**:

- In the table, check the **Use It** box for each key you want to include in the sequence (e.g., `F1`, `F6`).

2. **Set Repetitions**:

   - In the "Repetitions" field, enter the number of times to repeat the sequence (e.g., 2).

3. **Check Keys**:

   - Click the **Check Keys** button.
   - The application will execute all keys with **Use It** enabled, in the order they appear in the table, for the specified number of repetitions.
   - For each key:
     - The arm moves to the FI, SE, TH angles.
     - If **Click** is enabled, the FO motor performs a click.
     - Waits 1 second before moving to the next key.
   - The status bar will show "Sent: FI:,SE:,TH:" for each key.

Deleting a Key

1. **Select a Key**:

   - In the table, find the key you want to delete (e.g., `Esc`).

2. **Delete**:

   - Click the **Del** button in the "Action" column.
   - The key will be removed from `keys.json`, and the table will update.
   - The status bar will show "Deleted Esc".

Scripts Tab

The **Scripts** tab allows you to create, edit, and run scripts that sequence key presses with specified delays.

Creating or Editing a Script

1. **Enter Script Details**:

   - In the "Command Key" field, enter a name for the script (e.g., `Hello World`).
   - In the "Script Value" field, enter the script sequence (e.g., `H(2)e(2)l(1)l(1)o(1)`).

- Format: `<key>(<delay>)<key>(<delay>)...`
- Example: `F1(2)F6(2)` means press `F1` (2-second delay), then `F6` (2-second delay).
- The keys (e.g., `F1`, `F6`) must exist in `keys.json`.

```
[
 {
  "name": "F1",
  "fi": 102,
  "se": 84,
  "th": 63,
  "fo": 10,
  "use_it": true,
  "click": true
 },
 {
  "name": "Esc",
  "fi": 122,
  "se": 120,
  "th": 120,
  "fo": 10,
  "use_it": false,
  "click": false
 },
 {
  "name": "F6",
  "fi": 88,
  "se": 75,
  "th": 85,
  "fo": 10,
  "use_it": true,
  "click": true
 }
]
```

2. **Add or Update the Script**:

- Click the **Add Script** button.
- If the script name doesn't exist, it will be added to `scripts.json`. If it already exists, it will be updated.
- The status bar will show "Added script: Hello World" or "Updated script: Hello World".
- The table below will update to show the new or updated script.

3. **Edit an Existing Script**:

    - In the table, find the script you want to edit (e.g., `Hello World`).
    - Click the **Edit** button in the "Action" column.
    - The script name and value will load into the input fields, and the status bar will show "Editing script: Hello World".
    - Adjust the name or value as needed, then click **Add Script** to save the changes.

Running a Script

1. **Select a Script**:

    - In the table, find the script you want to run (e.g., `Hello World` with value `F1(2)F6(2)`).

2. **Run the Script**:

    - Click the **Run** button in the "Action" column.
    - The application will execute the script:
        - For `F1(2)F6(2)`, it will:
            - Look up `F1` in `keys.json` and move the arm to its angles (e.g., FI:102, SE:84, TH:63, FO:10).
            - Wait 0.5 seconds for the motors to settle, then wait 2 seconds (the delay).
            - Look up `F6` and move the arm to its angles (e.g., FI:88, SE:75, TH:85, FO:10).
            - Wait 0.5 seconds, then wait 2 seconds.
    - The status bar will show "Running script: Hello World", followed by "Finished script: Hello World".

Deleting a Script

1. **Select a Script**:

    - In the table, find the script you want to delete (e.g., `F1F6`).

2. **Delete**:

    - Click the **Del** button in the "Action" column.
    - The script will be removed from `scripts.json`, and the table will update.
    - The status bar will show "Deleted script: F1F6".

Listen Mic Tab

The **Listen Mic** tab allows you to execute scripts using voice commands by speaking their names.

Starting Voice Recognition

1. **Start Listening**:
    - Click the **Start Listening** button.
    - The status bar will show "Microphone activated", and the "Status" field in the tab will change to "Status: Listening...".
    - The application will begin listening for voice commands.

Using Voice Commands

1. **Speak a Script Name**:

    - Speak the name of a script from `scripts.json` clearly (e.g., "Hello World").
    - The "Last Heard" field will update to "Last Heard: Hello World".
    - The status bar will show "Recognized: Hello World", then "Matched script: Hello World", and "Running script: Hello World".

2. **Script Execution**:

    - If the spoken phrase matches a script name, the script will execute as described in the "Scripts Tab" section.
    - For example, if "Hello World" is linked to `F1(2)F6(2)`, the robotic arm will press `F1`, wait 2 seconds, then press `F6`, wait 2 seconds.
    - The status bar will show "Finished script: Hello World" when complete.

3. **Non-Matching Phrase**:

    - If the phrase doesn’t match any script (e.g., "Good Morning"), the status bar will show "No script found for: Good Morning".

Stopping Voice Recognition

1. **Stop Listening**:
    - Click the **Stop Listening** button.
    - The status bar will show "Microphone deactivated", and the "Status" field will change to "Status: Inactive".
    - The application will stop listening for voice commands.

Troubleshooting

General Issues

- **Application Doesn’t Start**:
    - Ensure all dependencies are installed (see `INSTALL.md`).
    - Run the application from the command line to see error messages:

      python main.py

- **Status Bar Shows Errors**:
    - Read the status bar messages for clues (e.g., "Not connected to any port", "Script not found: Hello World").
    - Refer to the specific troubleshooting sections below.

Keys Tab Issues

- **Arduino Not Connecting**:
    - Ensure the Arduino is plugged in and the correct COM port is selected.
    - Verify that the Arduino sketch can handle commands like `FI:90,SE:45,TH:10,FO:180`.
    - On Linux/macOS, you may need to grant permissions to the serial port:
        - Linux: `sudo chmod 666 /dev/ttyUSB0`
        - macOS: Ensure your user has access to `/dev/tty.*` devices.

- **Arm Doesn’t Move**:
    - Check the motor angles; values outside the valid range (0-200) may cause issues.
    - Ensure the Arduino is connected and responding to commands.
    - Test the angles manually using the **Test Angles** button.

- **Get Actual Data Fails**:
    - Ensure the Arduino sketch supports the `GET_ANGLES` command and returns angles in the format `90,45,10,180`.

Scripts Tab Issues

- **Script Fails to Run**:
    - Check the script format in the "Script Value" field (e.g., `F1(2)F6(2)`).
    - Ensure the keys used in the script (e.g., `F1`, `F6`) exist in `keys.json`.
    - Open `keys.json` and `scripts.json` to verify their contents:
        - `keys.json` should have entries like `{"name": "F1", "fi": 102, ...}`.
        - `scripts.json` should have entries like `{"name": "Hello World", "value": "F1(2)F6(2)"}`.

- **Invalid Script Format**:
    - The status bar may show "Invalid script format". Ensure the script value follows the format `<key>(<delay>)<key>(<delay>)...`.
    - Example: `F1(2)F6(2)` is valid; `F1(2)F6` is not.

Listen Mic Tab Issues

- **Voice Not Recognized**:
    - Ensure your microphone is working and set as the default input device.
    - Speak clearly and minimize background noise.
    - Check your internet connection (Google Speech Recognition requires internet).
    - If the status bar shows "Could not understand audio", try speaking louder or closer to the microphone.

- **Script Not Executed**:
    - Ensure the spoken phrase matches a script name in `scripts.json` exactly (case-insensitive).
        - Example: Saying "hello world" will match a script named "Hello World".
    - Verify that the script's keys are defined in `keys.json`.

- **Speech Recognition Error**:
    - If the status bar shows "Speech recognition error", check your internet connection.
    - For offline use, the application would need to be modified to use an offline speech recognition engine like CMU Sphinx.

Additional Tips

- **Backup Configuration Files**:
    - Regularly back up `keys.json` and `scripts.json` to avoid losing your configurations.
- **Customizing Scripts**:
    - Create scripts for common tasks (e.g., typing a phrase like "Hello World") by defining the sequence of keys and delays.
- **Voice Command Best Practices**:
    - Use short, distinct script names for voice commands (e.g., "Hello" instead of "Execute Hello World Sequence").
    - Test voice commands in a quiet environment for best results.

Support

For further assistance, refer to `INSTALL.md` for setup issues, or contact the developer with details of your issue, including any error messages shown in the status bar.