Node-Red MG400 Documentation

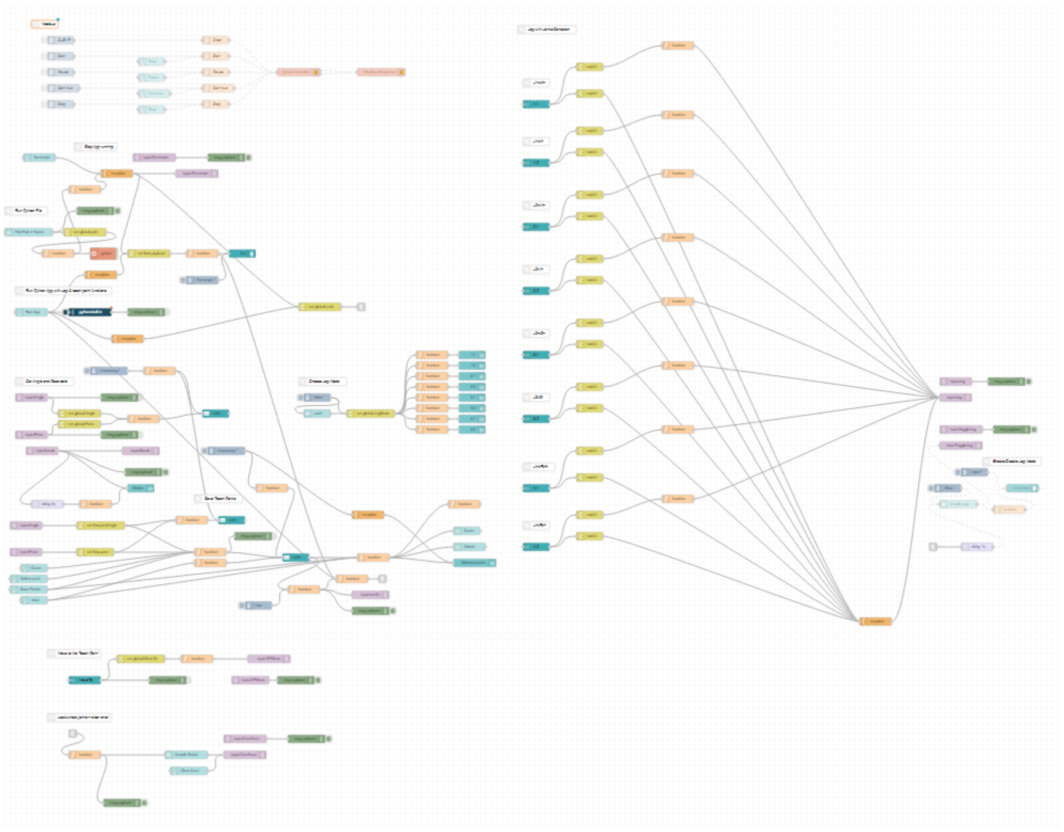
Contents

Note: For a detailed step-by-step guide, please refer to the documentation video for the CR5 Dobot Control found in the Github Repository Wiki. Although the video was created for the CR5, the only differences as of 23/2/2022 are:

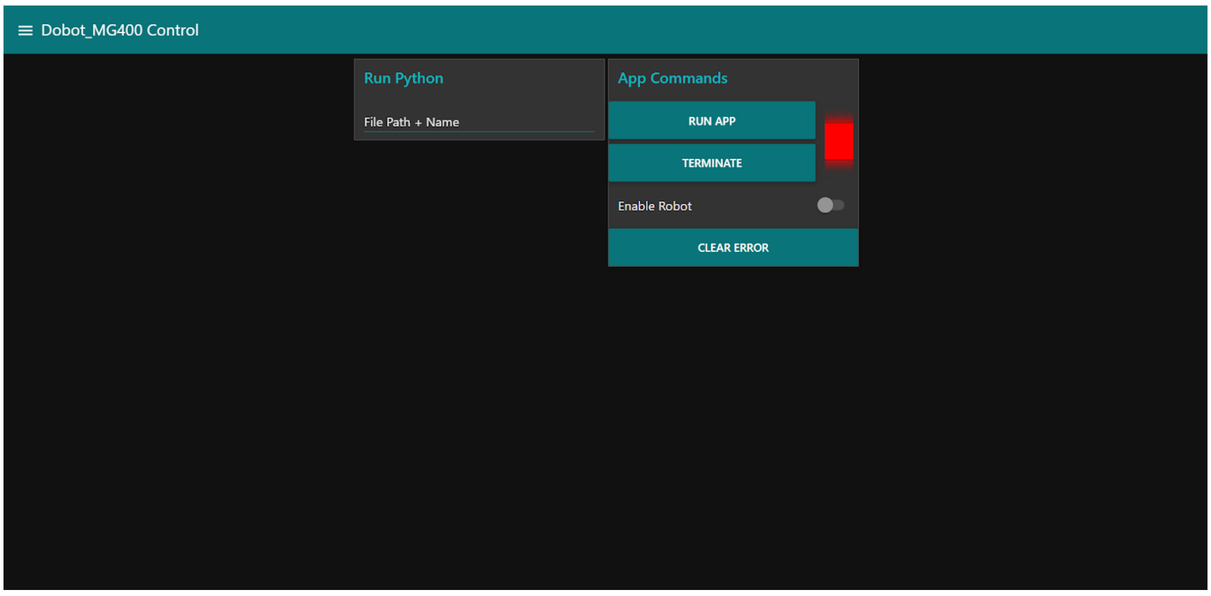
1. 4 Joint Angles and Cartesian Coordinates are used instead of 6
2. No Home tab
3. No Blockly implementation
4. No Modbus implementation
5. No Import and Export data Implementation

CR5 Video Documentation:
https://github.com/msf4-0/Collaborative-Robotics-Interface-CRI-/wiki

Introduction



1. The Node-Red Interface was created as a similar interface to the DobotDCStudio.
2. The Node-Red interface is able to control the Dobot MG400 using the Dobot MG400 Python API through MQTT connection.
3. The Interface controls are visible on a Node-Red dashboard.

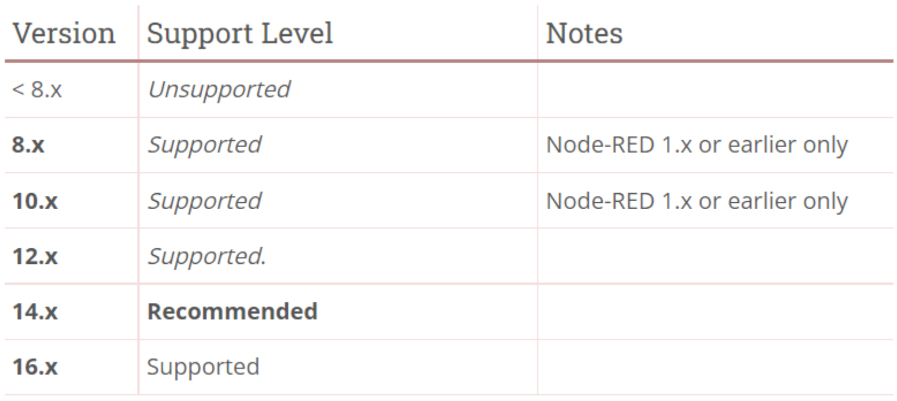


4. The Node-Red interface contains the following MG400 functions:
    - Enable and disable functionalities
    - Jog functionalities
    - Teach point functionalities

Node-Red Interface Setup

Node-Red installation

1. To set up the Interface, install node-red on your PC.
   Prerequisite: You need to have installed Node.js of the recommended versions below.

Version	Support Level	Notes
< 8.x	*Unsupported*	
8.x	*Supported*	Node-RED 1.x or earlier only
10.x	*Supported*	Node-RED 1.x or earlier only
12.x	*Supported.*	
14.x	**Recommended**	
16.x	Supported	

   To install Node.js, use the following link:
   https://github.com/jasongin/nvs/releases
   And download the latest version of the MSI installer.

2. To install on Windows, open Command Prompt on your PC and enter the following:
   ```
   npm install -g --unsafe-perm node-red
   ```

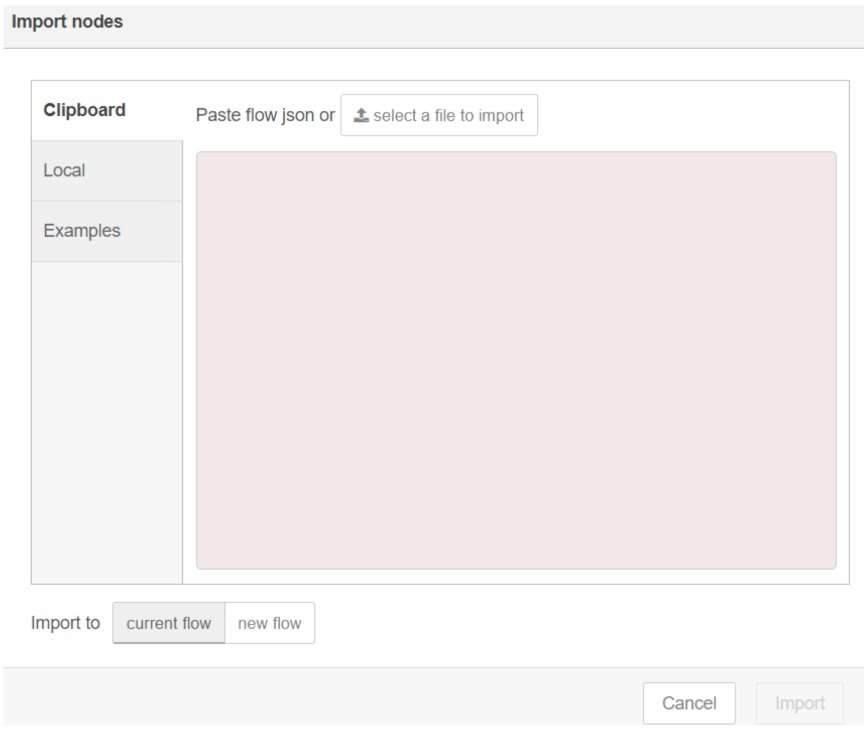
3. Once installation is complete, run Node-Red on your PC with the following:
   ```
   Node-red
   ```
   *Closing the Command Prompt window will end the Node-Red Session

4. Access the editor by entering the following into your browser:
   ```
   http://<ip-address>:1880
   ```
   OR
   ```
   http://localhost:1880/
   ```

5. Next, navigate to Command Prompt again and install the required palette extensions:
   ```
   npm install node-red-contrib-mqtt-broker

   npm install node-red-contrib-pythonshell

   npm install node-red-contrib-ui-led

   npm install node-red-dashboard

   npm install node-red-node-ui-table
   ```

   These palettes are used in the Node-Red Dobot Interface. Ensure they are installed before opening the node-red flow file.

6. To open the Dobot Interface file on Node-Red, navigate to the three lines on the top right corner of Node-Red and select “Import”.
7. Click “select a file to import” then choose the MG400_NodeRed and MG400_csvflow files and click Import.
   The flows will be imported onto the node-red window.



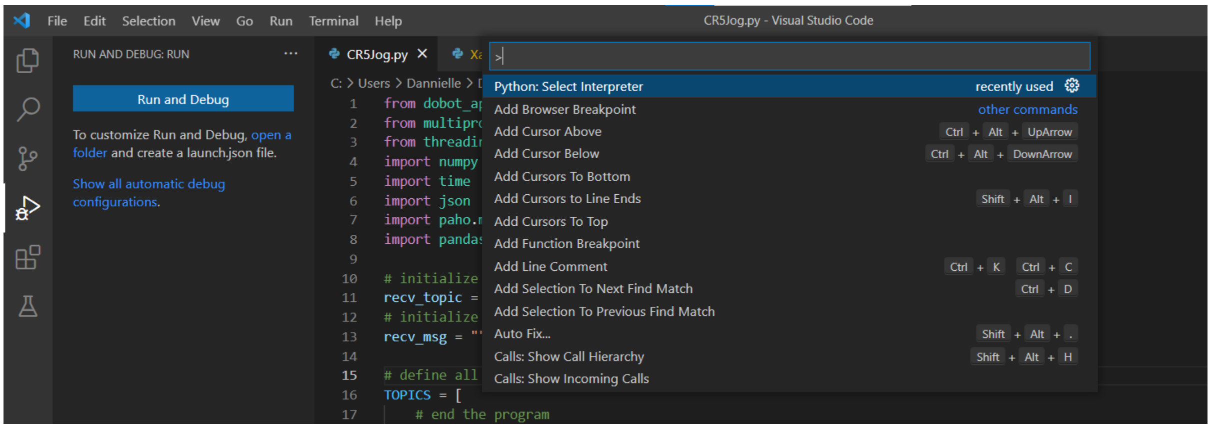
<u>Python Editor setup</u>

1. To open the Dobot Python API files, you will need a source-code editor that can support Python.
   Eg: Visual Studio Code, VScodium

2. To download Visual Studio Code, use the following link:
   https://code.visualstudio.com/download
   Click the download button according to your supported device.
   Open Visual Studio Code and open the Python API file on the application.

3. Install the Python extension using the following link:
   https://marketplace.visualstudio.com/items?itemName=ms-python.python

4. Activate the Python extension using the keys: **CTRL+Shift+P**
   Then enter or select the following **Python: Select Interpreter**



To customize Run and Debug, open a folder and create a launch.json file.

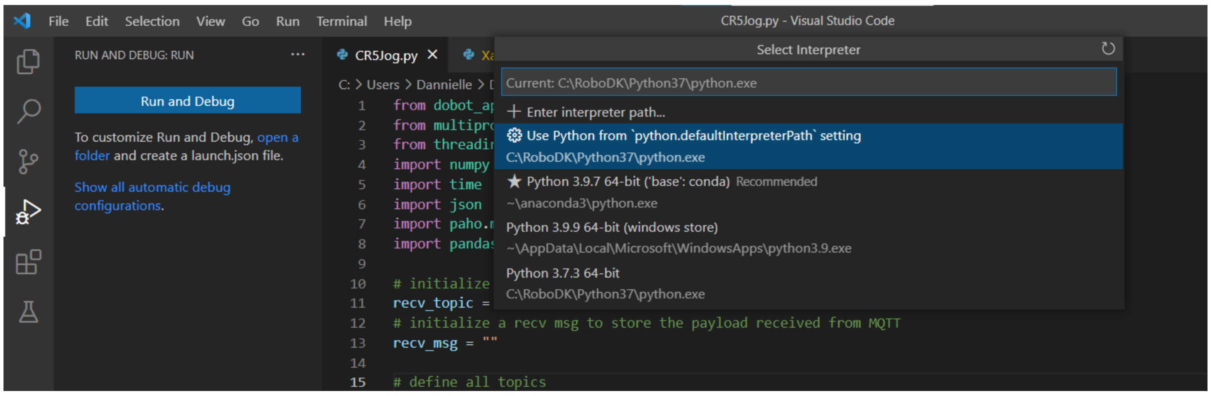
Show all automatic debug configurations.

C: > Users > Dannielle >

```
from dobot_ap
from multipro
from threadi
import numpy
import time
import json
import paho.
import panda

# initialize
recv_topic =
# initialize
recv_msg = "

# define all
TOPICS = [
    # end the program
```

```
>|
Python: Select Interpreter                      recently used
Add Browser Breakpoint                          other commands
Add Cursor Above                    Ctrl + Alt + UpArrow
Add Cursor Below                    Ctrl + Alt + DownArrow
Add Cursors To Bottom
Add Cursors to Line Ends            Shift + Alt + I
Add Cursors To Top
Add Function Breakpoint
Add Line Comment                    Ctrl + K  Ctrl + C
Add Selection To Next Find Match    Ctrl + D
Add Selection To Previous Find Match
Auto Fix...                         Shift + Alt + .
Calls: Show Call Hierarchy          Shift + Alt + H
Calls: Show Incoming Calls
```

5. A dropdown box will appear and display the interpreters available. Select an available Python interpreter.



To customize Run and Debug, open a folder and create a launch.json file.

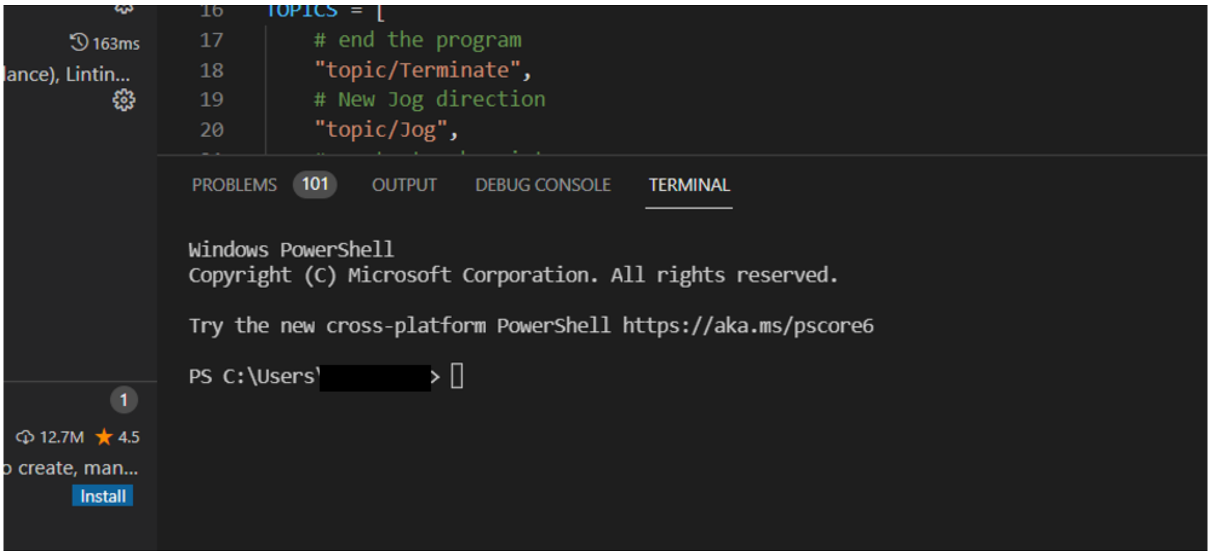
Show all automatic debug configurations.

C: > Users > Dannielle >

```
from dobot_ap
from multipro
from threadi
import numpy
import time
import json
import paho.
import panda

# initialize
recv_topic =
# initialize a recv msg to store the payload received from MQTT
recv_msg = ""

# define all topics
```

```
Current: C:\RoboDK\Python37\python.exe
+ Enter interpreter path...
Use Python from `python.defaultInterpreterPath` setting
C:\RoboDK\Python37\python.exe
★ Python 3.9.7 64-bit ('base': conda) Recommended
~\anaconda3\python.exe
Python 3.9.9 64-bit (windows store)
~\AppData\Local\Microsoft\WindowsApps\python3.9.exe
Python 3.7.3 64-bit
C:\RoboDK\Python37\python.exe
```

6. Next, install the following extensions which will be used in the Python API file.
   Enter the following in the terminal window:
   pip install numpy
   pip install paho-mqtt



```
TOPICS = [
    # end the program
    "topic/Terminate",
    # New Jog direction
    "topic/Jog",
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\[redacted]>
```

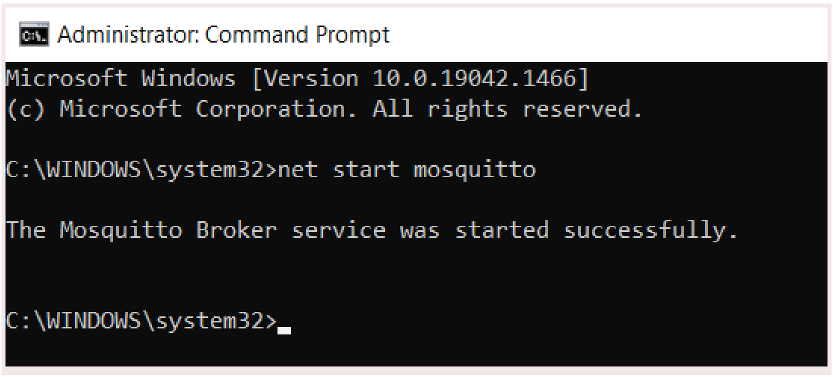
MQTT Setup and Installation

1. In this Node-Red interface, an MQTT connection is used to send and receive data from the Python API script.

2. To set up the MQTT connection, you will need to install an MQTT broker. Use the following link:
   https://mosquitto.org/download/
   Download the 64-bit Windows version onto your PC.

3. Once the download is complete, install the broker and complete the installation setup process.

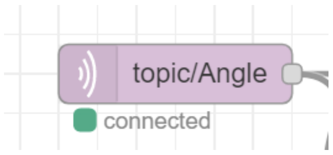
4. To start up the Mosquitto MQTT service, run Command Prompt as **Administrator** and enter the following:
   `net start mosquitto`

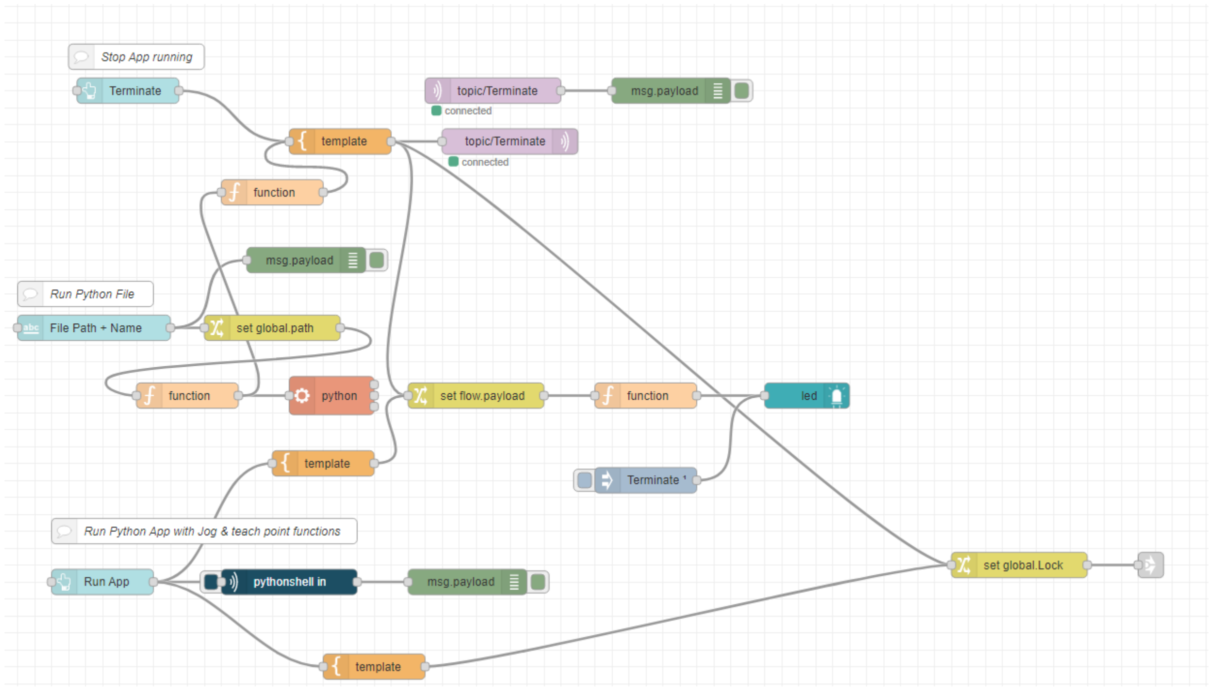
   ```
   Administrator: Command Prompt
   Microsoft Windows [Version 10.0.19042.1466]
   (c) Microsoft Corporation. All rights reserved.

   C:\WINDOWS\system32>net start mosquitto

   The Mosquitto Broker service was started successfully.


   C:\WINDOWS\system32>_
   ```

   You may stop the service using the following command:
   `net stop mosquitto`

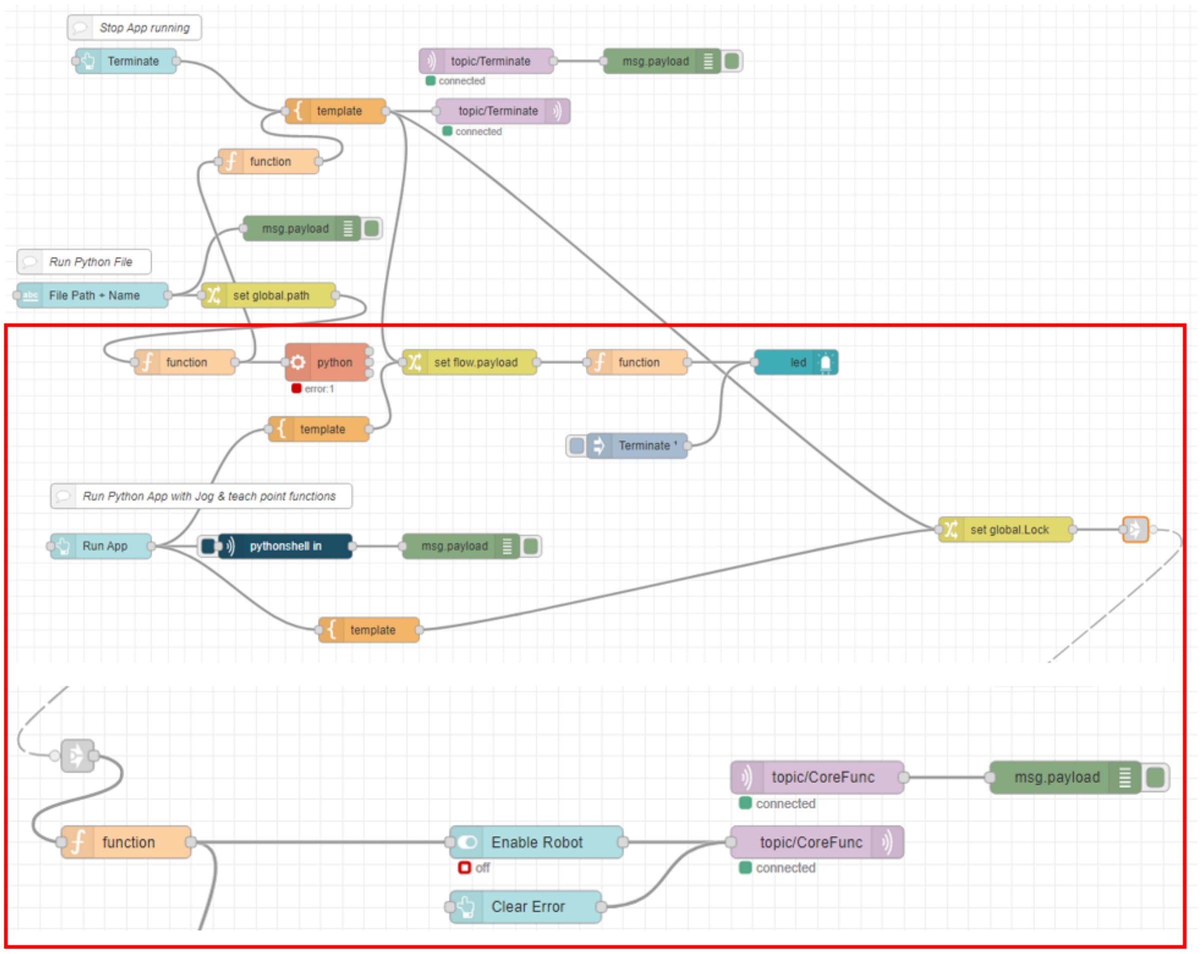
5. When the MQTT connection is setup the square under the MQTT nodes will turn green and show “Connected”.

   Eg: 

Python Shell (Enable/Disable Robot)



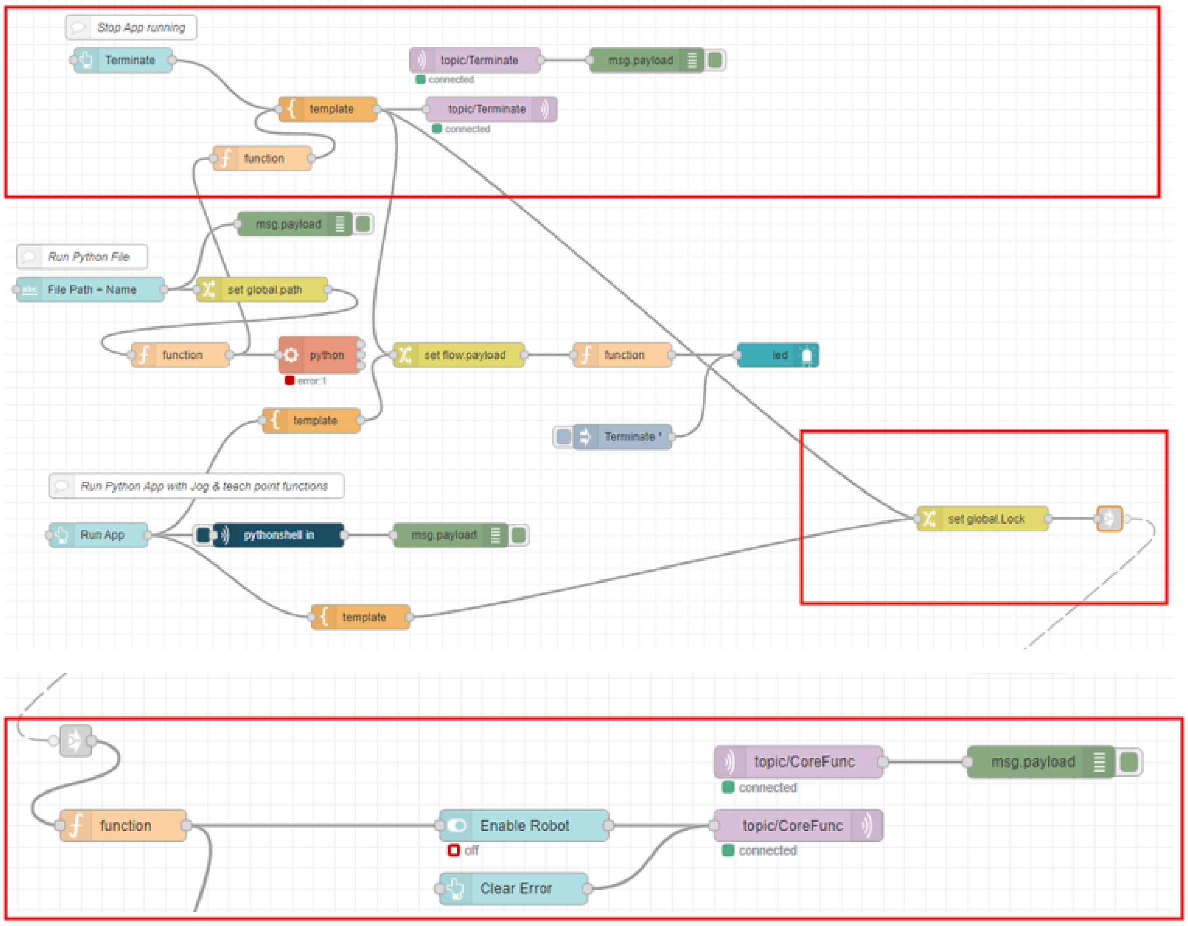
1. Run Python App with Jog and Teach Point function



   a. Clicking the Run App button on the UI dashboard, will:
      i. Trigger the python shell node to run the Dobot API Python Script

ii. Send a msg.payload to turn on the ui LED.
iii. Send an enable msg.payload to a function that determines whether to turn on or turn off the enable toggle UI.
iv. When the toggle is on, a message is sent through MQTT to the Python script which will run the python enable robot function.

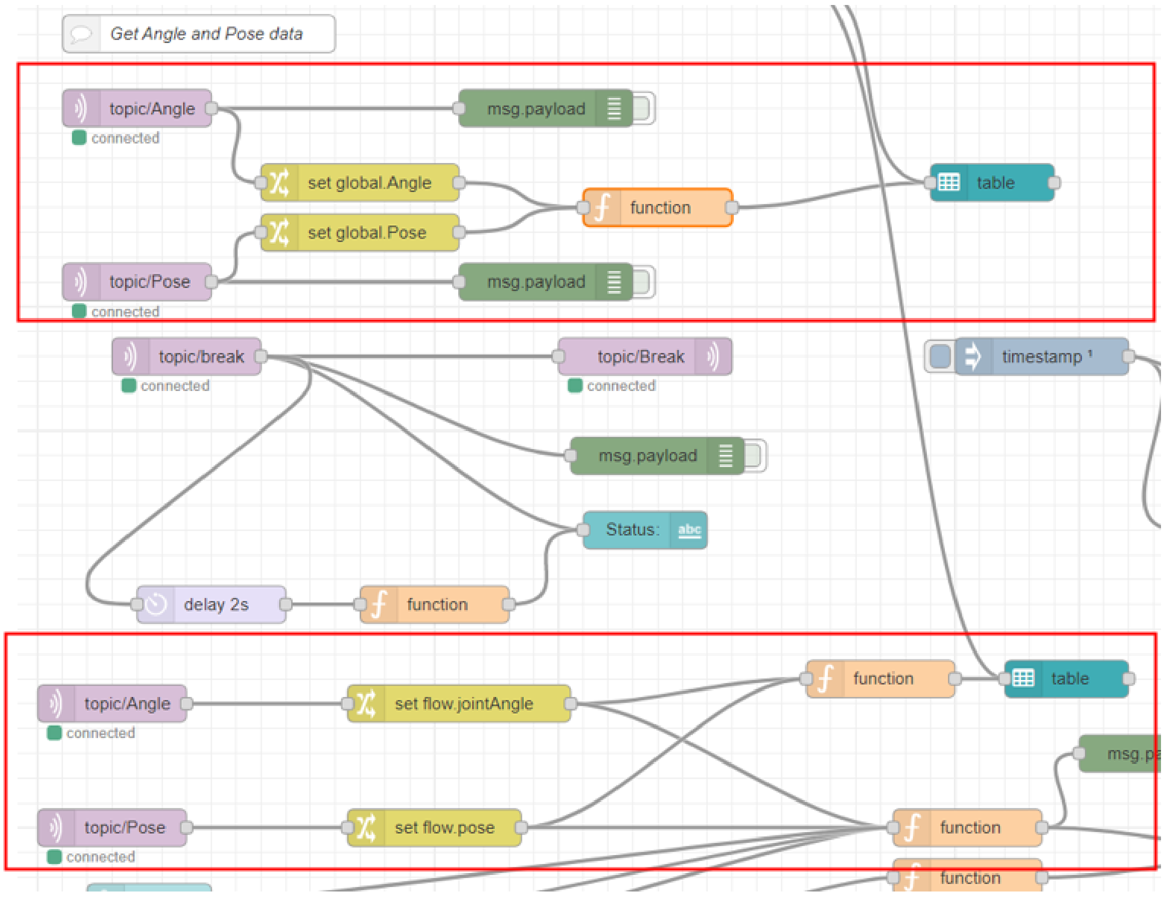
2. Stop App running



a. Clicking on the terminate button will:
   i. Cause a function to send a terminate msg.payload to the Python script through MQTT.
   The message sent to the Python Script through MQTT will enter a terminate function to disable the robot and terminate the program.

   ii. Send an terminate msg.payload to a function that determines whether to turn on or turn off the enable toggle UI.
   When the toggle is off, no message is sent through MQTT to the Python script.

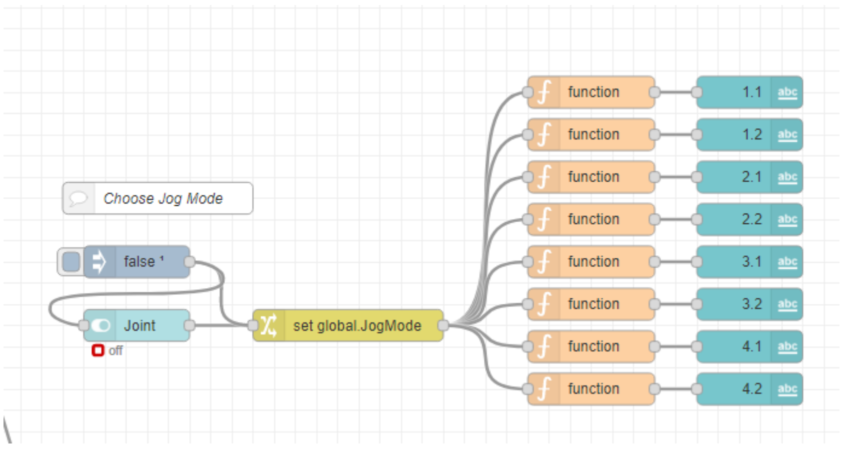
Get Pose and Joint Angles

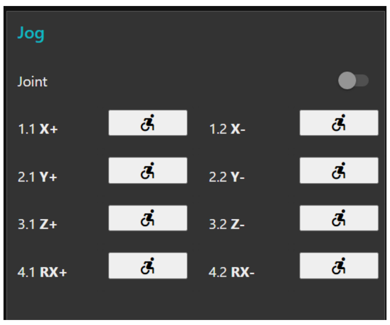


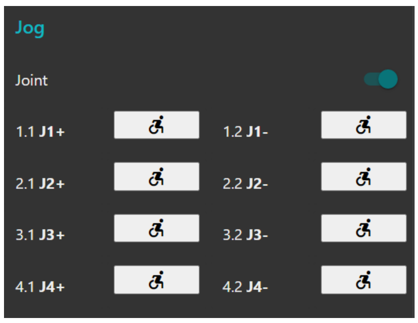
1. Joint angles and Pose coordinates are obtained from a Python function in the Python script and sent to Node-Red through MQTT.
2. The data is sent to a function node to sort out the data as an object to be displayed in a table UI.
3. The live joint angles are displayed in 2 table UIs. One in the jog tab and one in the teach points tab.

<u>Jog Interface</u>

1. Jog Labels

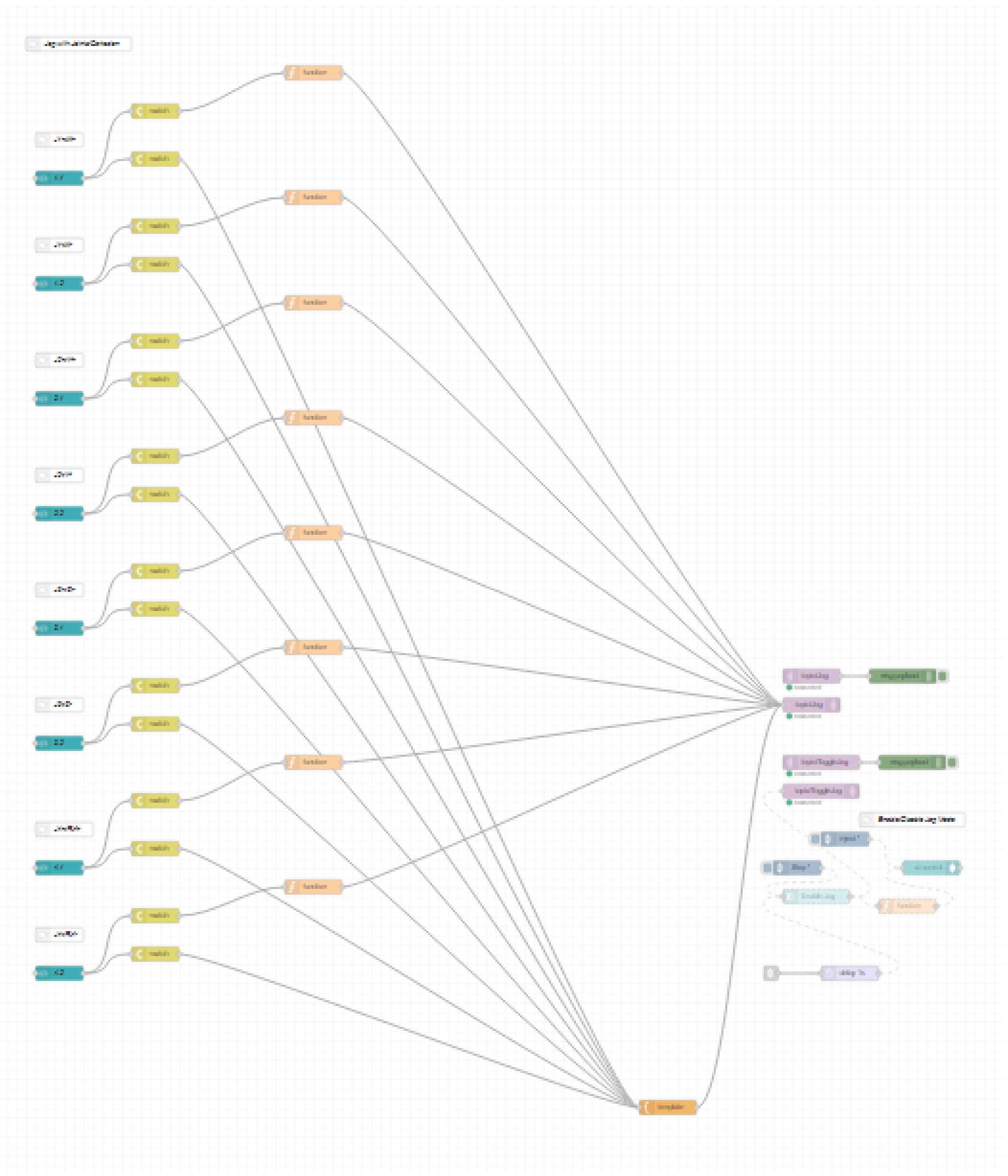


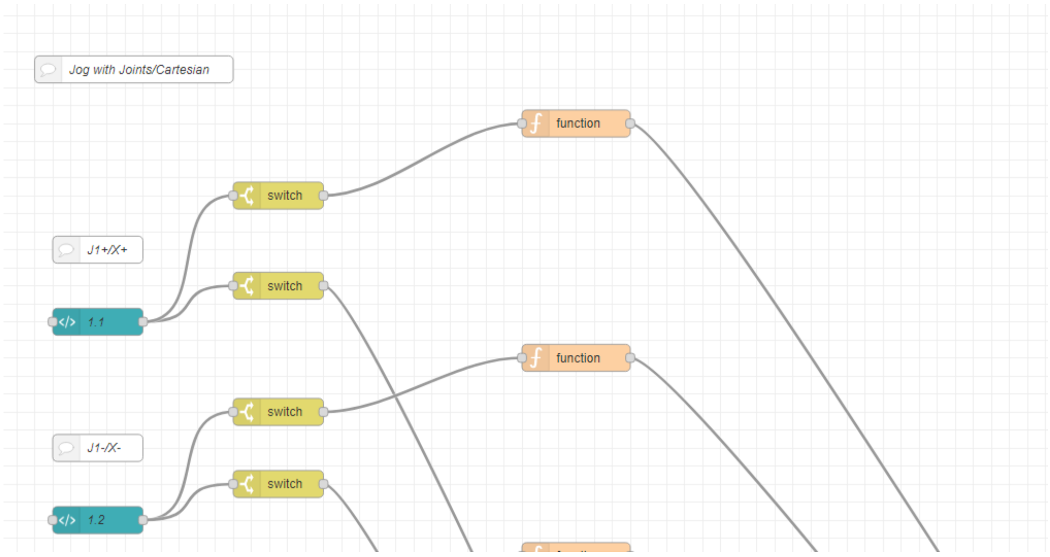




   a. The Jog mode toggle which is initially switched off will display the Jog panel with Cartesian Coordinate labels. When the toggle is switched on, the dashboard will display the Jog panel with Joint Angle labels.

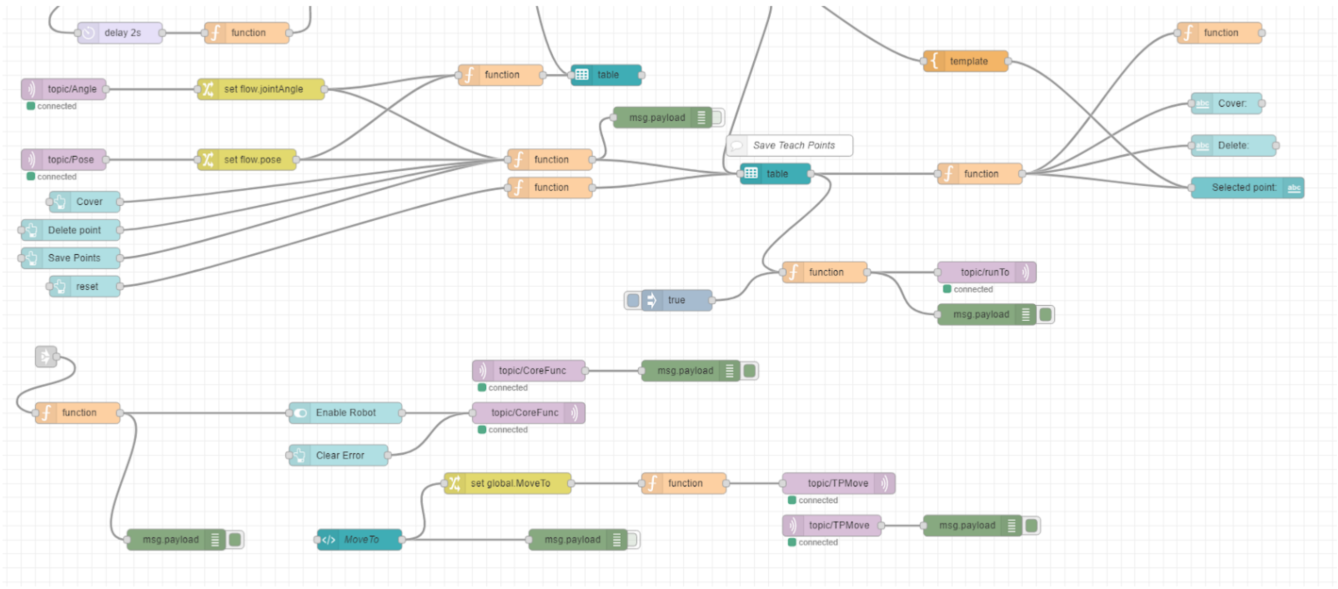
2. Jog buttons

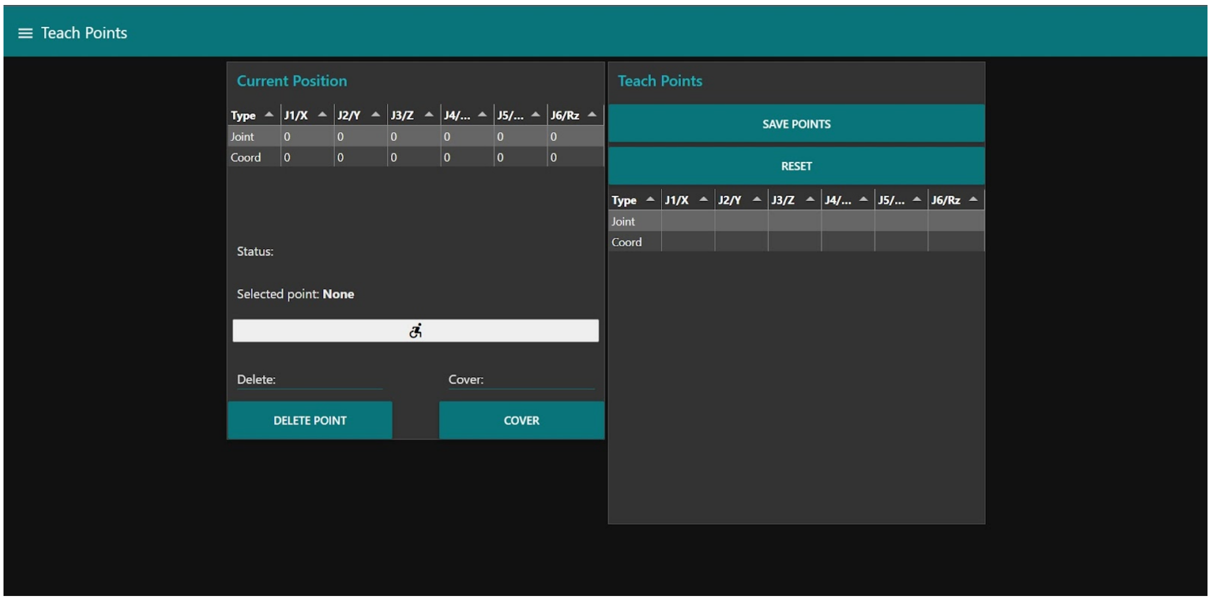




a. Clicking on the Jog button on the Jog panel will send an increment angle/pose syntax (depending on the jog mode) to the Python Script through MQTT.
b. If the joint mode is on then the syntax sent will have the joint increment syntax.
c. If the cartesian coordinate mode is on then the syntax sent will have the cartesian angle increment syntax.
d. The increment angle/pose syntax that is sent to the Python script will cause the robot to jog in that direction until the jog button is released.

Teach Point Interface




Teach Points

Current Position

Type	J1/X	J2/Y	J3/Z	J4/...	J5/...	J6/Rz
Joint	0	0	0	0	0	0
Coord	0	0	0	0	0	0

Status:

Selected point: **None**

Delete:

Cover:

DELETE POINT

COVER

Teach Points

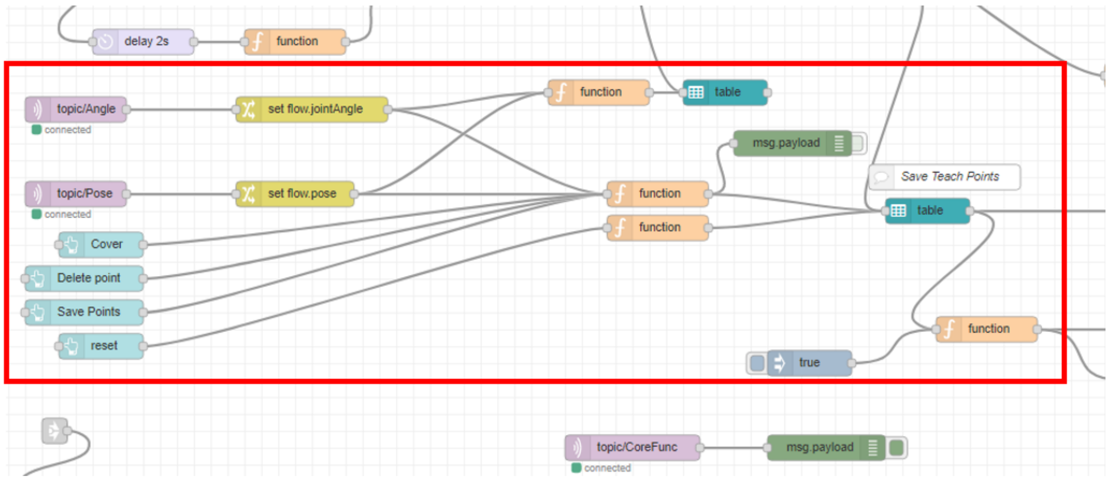
SAVE POINTS

RESET

Type	J1/X	J2/Y	J3/Z	J4/...	J5/...	J6/Rz
Joint						
Coord						

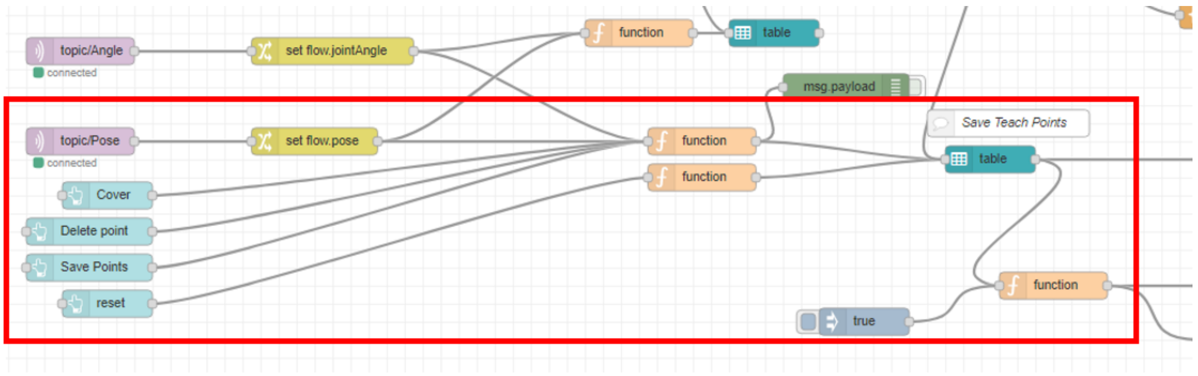
The teach point tab contains 4 features:

1. Save Teach Points



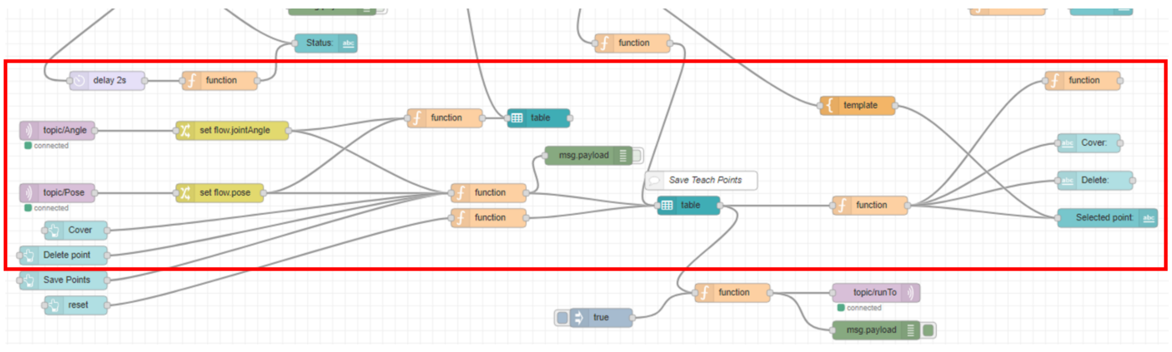
   a. Clicking the Save Points button triggers the current pose and joint angles to be sent from the Python script through MQTT to a function node.
   b. The function node converts the coordinates into an object format and stores them in an array. The saved object element is displayed on the Saved Points table.

2. Reset Teach Points



   a. Clicking the Reset button will trigger a reset msg.payload to be sent to a function node which will delete all elements in the stored array.
   b. This causes the table to be displayed as empty in the dashboard.

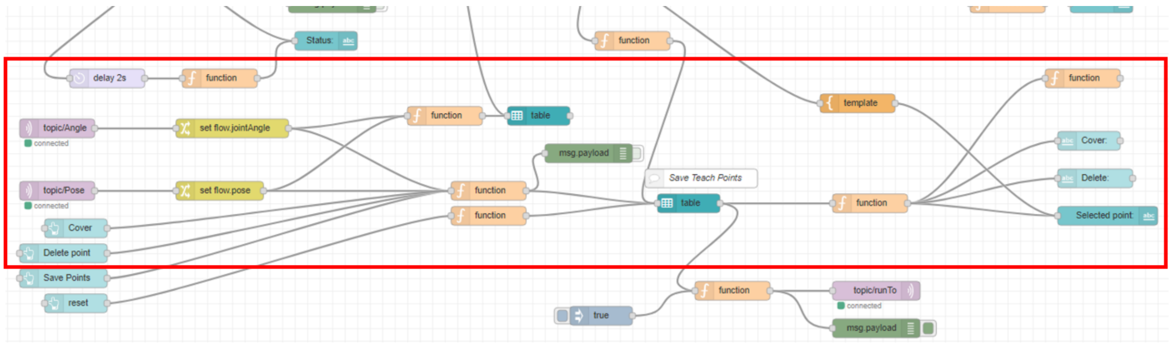
3. Delete Teach Point



   a. When a row in the Saved Points table is clicked, the clicked row number is passed through a function node for row number calculation.

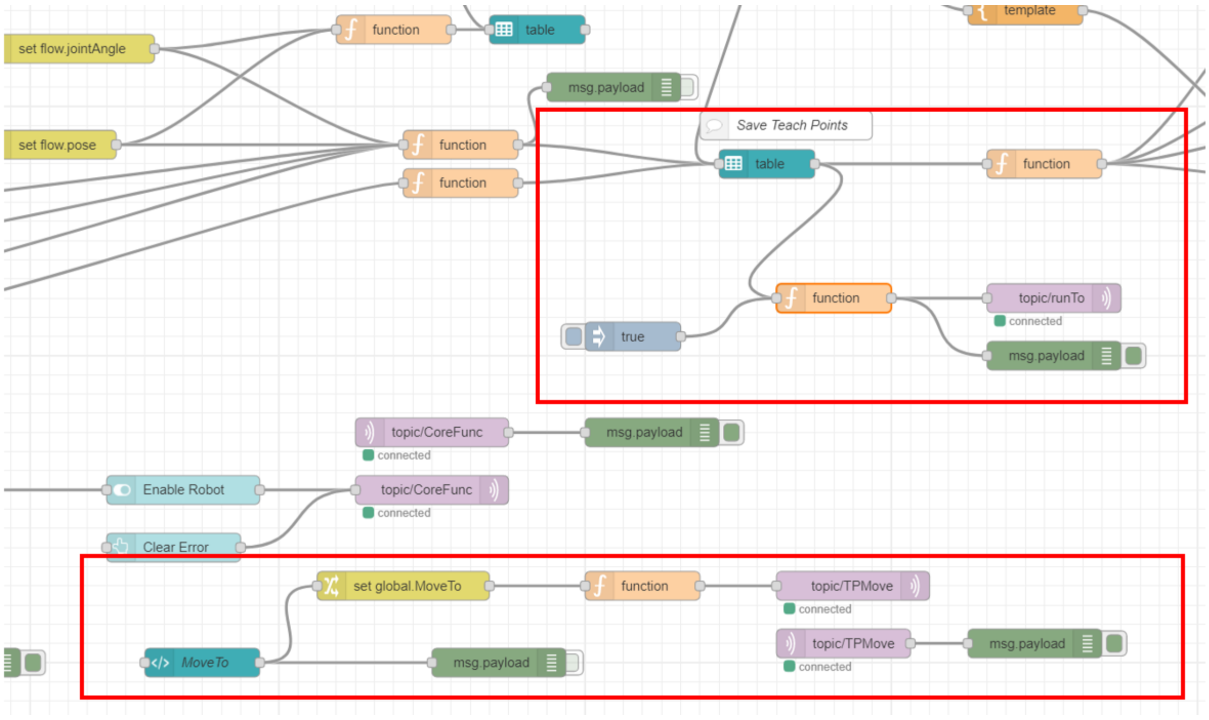
b. The calculated row number is displayed on the cover point and delete point text field.
c. When the delete button is clicked, a delete msg.payload is sent to a function node which will clear the object contents for the respective row but will not remove the object.
d. The empty element row will be displayed on the saved points table.

4. Cover Teach Point



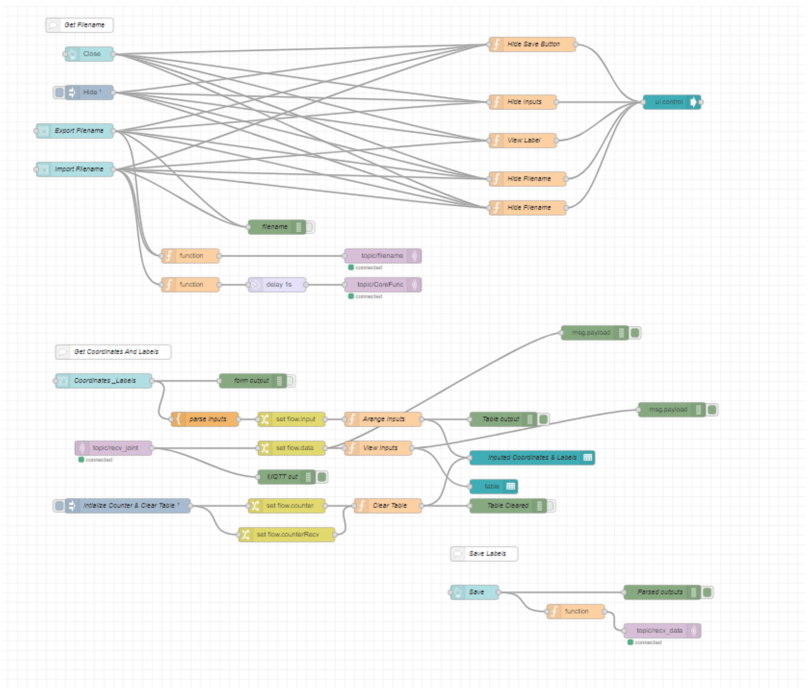
a. When a row in the Saved Points table is clicked, the clicked row number is passed through 2 function nodes. One for row number and the other for array element number calculation.
b. The calculated row number is displayed on the cover point and delete point text field.
c. When the cover button is clicked, a cover msg.payload is sent to a function node which will replace the object contents of the selected row with the current live coordinates.
d. The new covered coordinates will be displayed in the covered row in the saved points table.

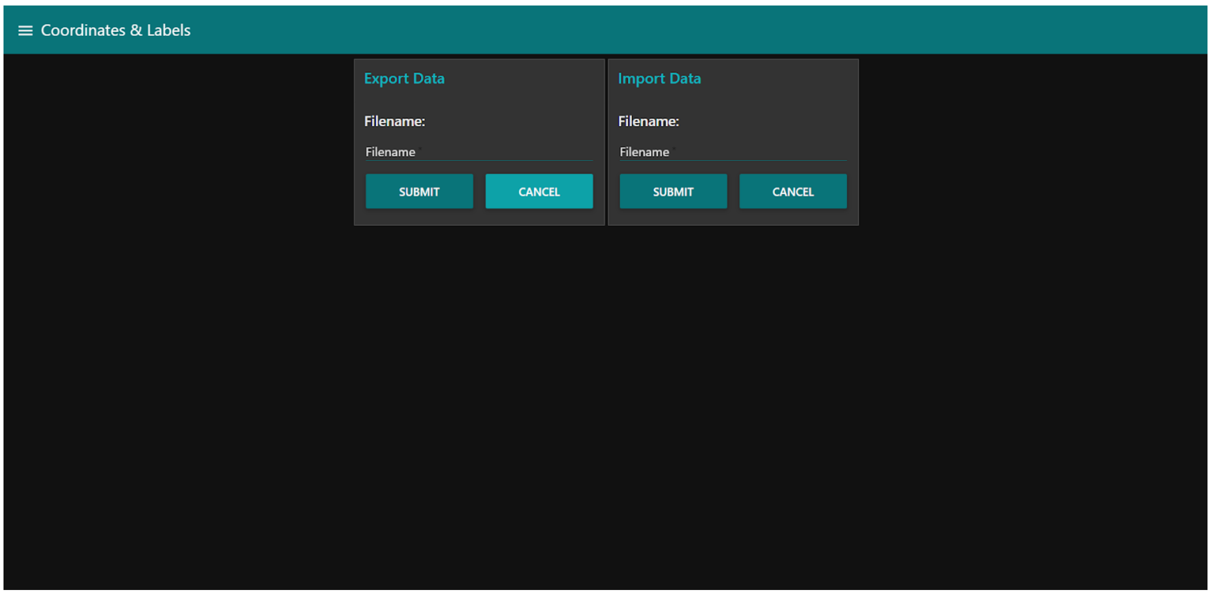
5. Move to Teach Point

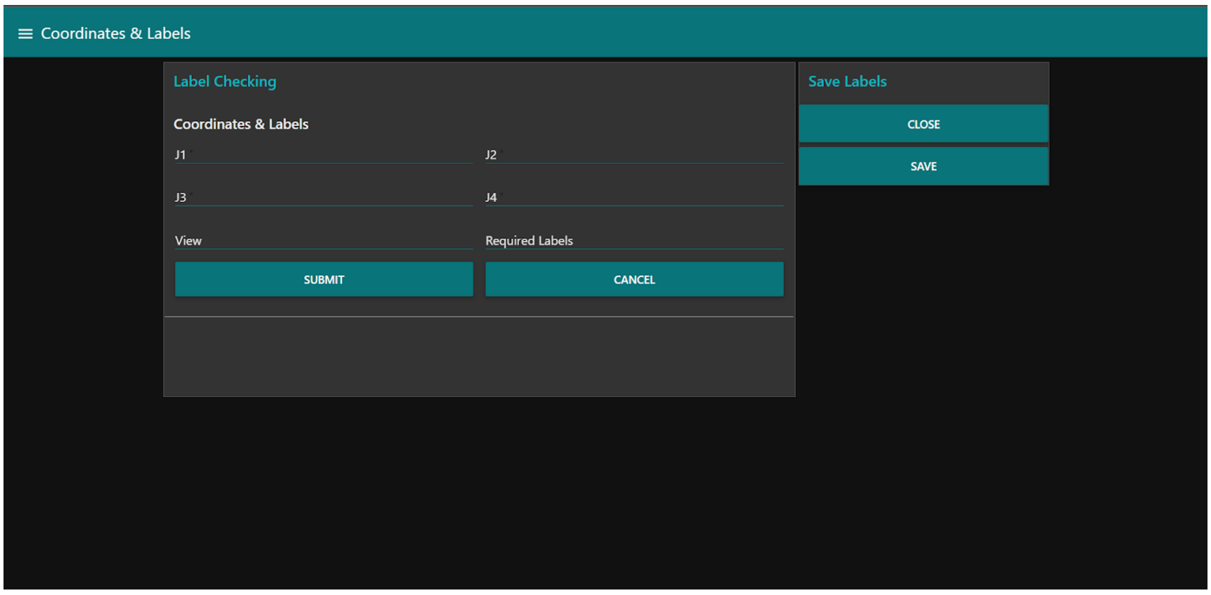


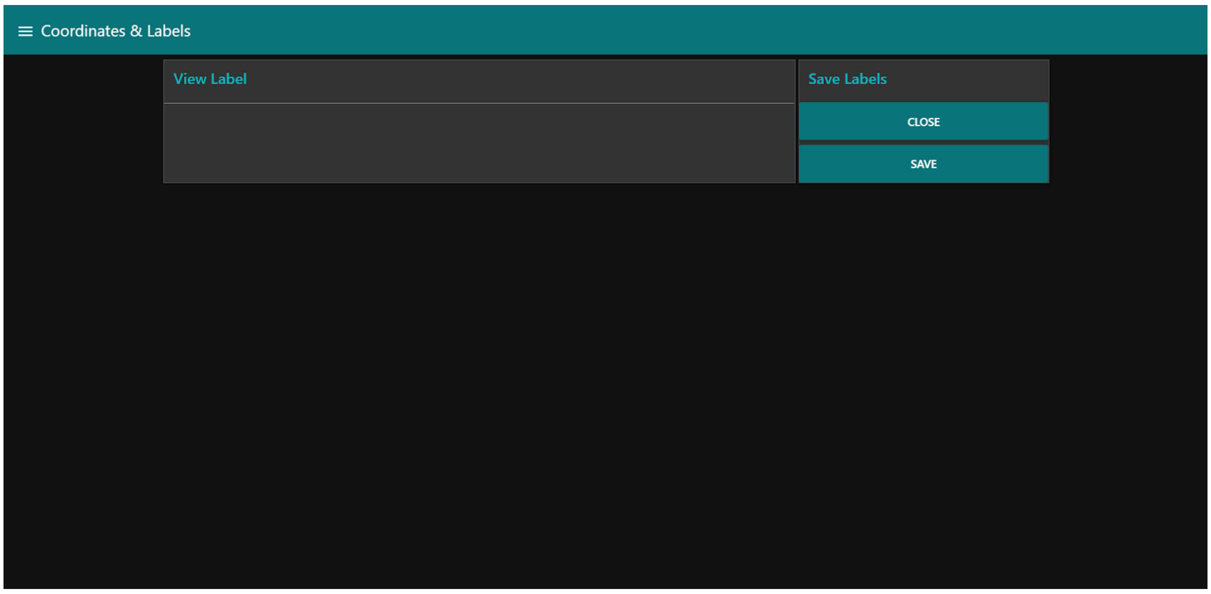
a. When a row in the Saved Points table is clicked, the clicked row number is passed through a function nodes that calculates the array element number and converts the selected object into an array of coordinates.
b. The array is passed to the Python script through MQTT as a string. The string is converted back to Python syntax inside the python script.
c. When the Move to teach point button is held down, a start message is sent to the Python Script through MQTT. This message will start jogging the robot to the respective teach point.
d. When the Move to teach point button is released, a stop message is sent to the Python Script instead. This message will stop the jog motion at any point of motion and keep the robot stationary.
e. The robot will continue to move when the move to button is held down again and will only stop when it has reached.

Import and Export Data



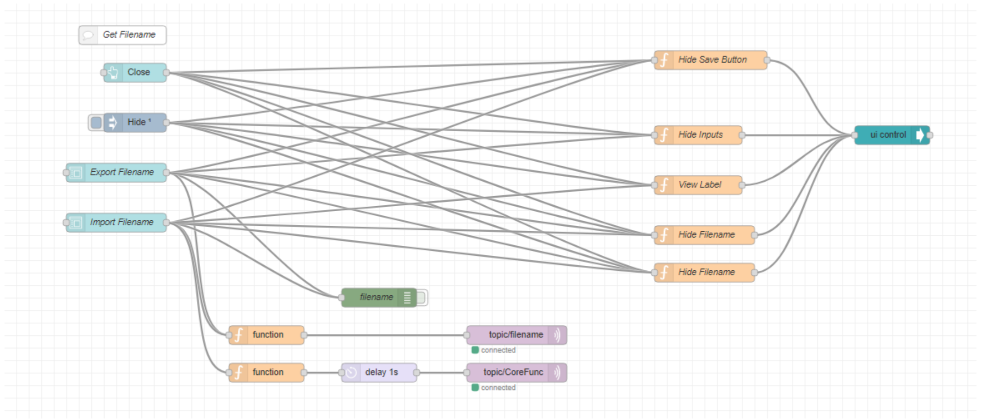






Coordinates & Labels tab has x features:

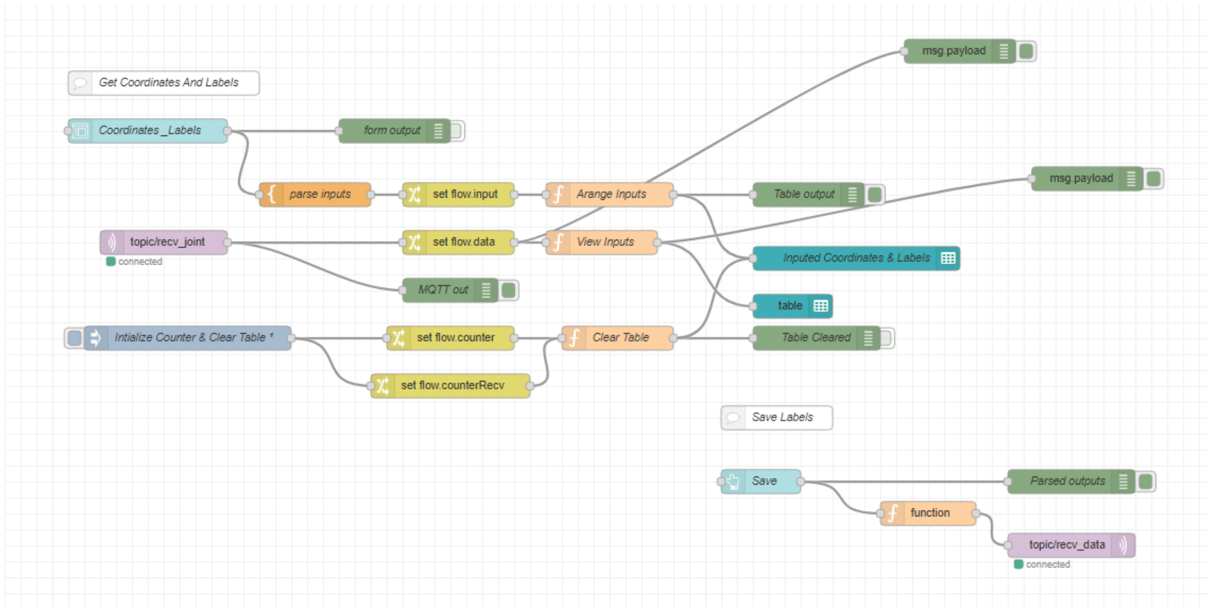
1. UI control



a. At default, it will have certain UI hidden from the user and others visible
b. When a filename is given under export or import form the UI will update accordingly.

c. Information of the filename to retrieve data from or create is sent to the python script via MQTT
d. Close button will close certain UI and enable others as necessary.
e. A home page is made from this UI, it does not serve much functionality

2. Creating the dataset



a. As of the current Node-Red App version (23/2/2022), this section of the Import Export data has not been implemented yet.
b. To implement similar features as the Import and Export data of the CR5 Node-Red App, please refer to the CR5 documentation, python script and video documentation on CSV. The implementation should be very similar except that four joint angles are used instead of six.
c. The only features available as of the (23/2/2022) version, is in the Export Data section, where you will be able to store the points that you input into the form, onto a table.