User Manual

Installation:

Python 3.7+ is required to be installed on the machine.

Additionally, 2 Python modules need to be installed: PySerial and Dear PyGui.

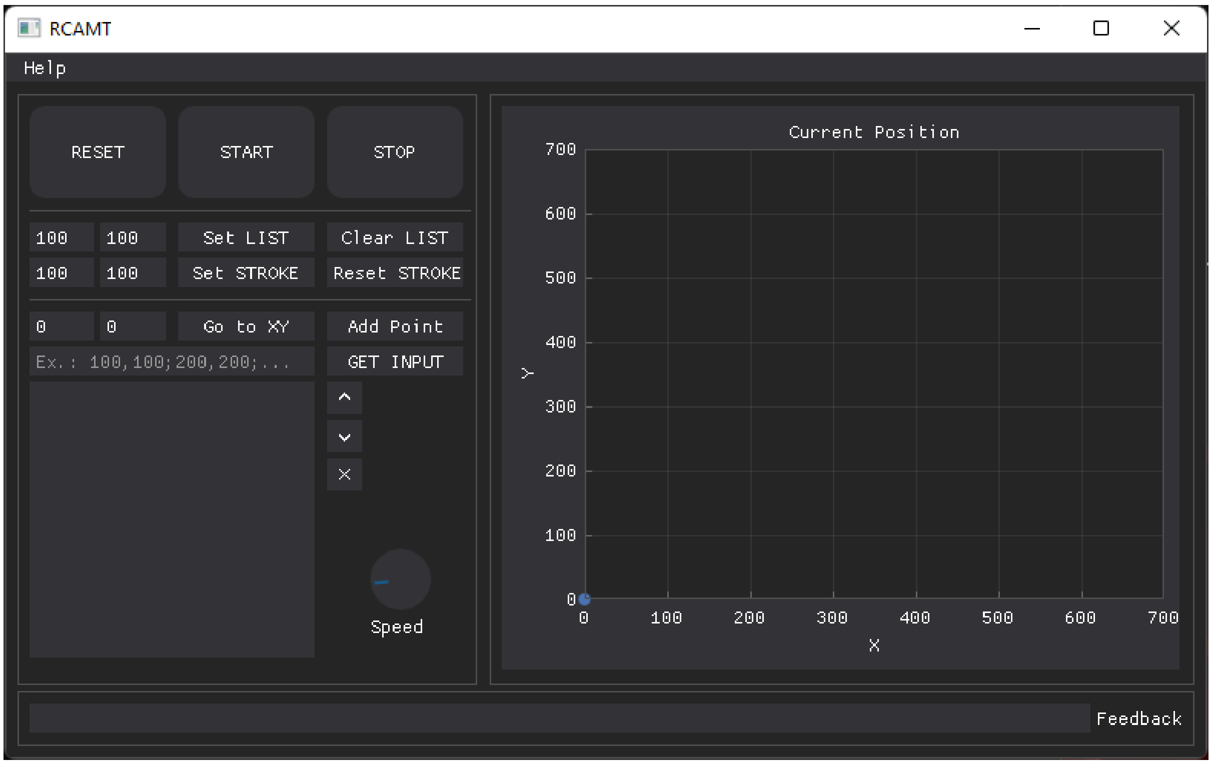
They can be installed with following commands:

- pip3 install pyserial
- pip3 install dearpygui

Launching:

With all modules installed, you should be able to run the application with the basic "python3 gui.py" command. Make sure that the device (Arduino UNO) is connected via USB before running the app.

If everything goes as planned, following user interface will appear:



The user interface is split in two - on the right side there is the tracking system that displays the robot's position in real time, on the left - the control panel that contains all of the buttons that allow user to manipulate the robot.

At the bottom there is the feedback - it simply displays any data that is sent back by the Arduino board.

Tooltips are available for each button. You can enable them at the top of the interface in Help > Enable Tooltips.

Usage:

1) Switch the power on in the control panel of the Arduino UNO (gray box at the bottom of the robot).
2) Make sure the red emergency button is not locked in.
3) Connect the Arduino to your PC via USB cable.
4) Launch the application by running “python3 gui.py” command in the project’s folder.
5) When the GUI appears, press RESET button, as it is necessary for the serial connection to initialize.

If you followed every step, you should be able to see the feedback display a long string with the data sent by Arduino. You can now freely interact with the application.

Troubleshooting:

Connection related problems:

Most of the errors will be linked to the USB connection. Clear indication of that is if the console outputs “Unable to find Arduino board with given SNR.”. It means the Arduino board is not connected to the PC, or not powered - verify the power switch and make sure the emergency stop button is not pressed.

Arduino related problems:

It is also possible someone has modified the code loaded into Arduino board. If it is the case, download Arduino IDE, connect to the Arduino UNO board and load the RobotControl-2axis.ino file that can be found in backup > arduino > RobotControl-2axis.

Module version related problems:

It is also possible that the modules used in the development may have changed. Some functionalities may become obsolete etc. In this case, download exact versions of the two modules used in development and install them:

- PySerial 3.5
- Dear PyGui 1.4.0

To Future Devs

Hi. Here's a quick intro to the two modules you will be using (or not) while working on this project.

PySerial:

PySerial is a module used to establish a serial communication between the microcontroller (in this case Arduino Uno that you are using) and the PC. This module allows you to send a command to the Arduino in form of a string (chain of characters). The Arduino will interpret this string and will execute required action. Arduino will also return a string that contains info regarding the state of the robot (it's position, speed, stroke etc.).

To use this module, first, you need to connect to your device:

```
def connection():
    connect = find_port('95530343434351A012C1')

    return serial.Serial(connect, baudrate=115200, bytesize=serial.EIGHTBITS,
timeout=1, write_timeout=3)
```

```
connect = find_port('95530343434351A012C1')

ser = serial.Serial(connect, baudrate=115200, bytesize=serial.EIGHTBITS,
timeout=1, write_timeout=3)
if not ser.isOpen():
    ser.open()
```

This chunk of code allows you to connect to your Arduino. Connection can be made by specific USB port, or a Serial Number, as it is shown above.

When the connection is established, you can use write method to send a string to the Arduino:

```
ser.reset_input_buffer()
command = "GOTOPOSITION,350,350\n"
ser.write(command.encode())
```

Resetting input buffer is essential before writing, as well as finishing the command by a \n and encoding the command string by .encode() .

To read the output of the Arduino you can use readline method:

```
ser.reset_output_buffer()
feedback = ser.readline().decode('ascii')
```

It is also advised to reset the output buffer, although not always necessary.

When it comes to the actual commands, the Arduino is coded to recognize following commands:

- INITAXISORIGINS — sends robot to the 0,0 position and resets it.

- GOTOPOSITION,X,Y — sends robot to X,Y position.
- X — stops all motion.
- SETSPEED,X — sets robots movement speed to X.
- SETSTROKE,X,Y — limits the zone in which the robot can move around.
- BRKENGAGE — engages the breaks of the robot (doesn't stop movement).
- BRKRELEASE — releases the breaks.

Some remarks on Serial connection:

- Only 1 connection can be established at a time.
- You can't use threads to write or read in parallel.
- The feedback is sometimes only partial.

Dear PyGui:

DPG is a relatively easy to understand GUI library. You got different inputs and buttons that are identified by their tag (string). Any button presses triggers btn_callback method that contains all of the button's behavior.

Main loop can be found on gui.py:364.

```
while dpg.is_dearpygui_running():
```

Buttons callback method can be found at the top of the file:

```
def btn_callback(sender, app_data, user_data):
```

Sender – is a tag of the element that was triggered.

App data – is the data sent by said element.

User data – is the data specified by user, in our case it's not used.

That's pretty much it. If you have any questions, feel free to contact me on discord – Doberman#7170 or via email: grayspot@inbox.ru