**Craniobot**

**System requirements**

1. Software dependencies
   * Windows 10
   * 64 bit
   * Internet connection
2. This software has only been tested on windows 10 64 bit systems
3. Required non-standard hardware
   * See manuscript for custom hardware description
   * TinyG (~30-45 minutes)
     + Install hardware according to - <https://github.com/synthetos/TinyG/wiki/TinyG-Start> (you do not need a fan nor a program debugger)
     + Connect Tiny G according to - <https://github.com/synthetos/TinyG/wiki/Connecting-TinyG>
     + For custom notes on our installation see "Tiny G Seup.docx"

**Installation**

1. Download software (~10-15 minutes)
   1. Download python 2.7.14 for windows
      * Click windows x86-64 MSI installer located at <https://www.python.org/downloads/release/python-2714/>
      * Run the downloaded file
   2. Download arduino 1.8.1 windows
      * <https://www.arduino.cc/en/Main/Donate> click "just download" or donate if you wish to support arduino
      * Run the downloaded file
   3. Download zip folder of craniobot software from ".zip" or download repository from <https://github.com/bsbrl/craniobot>
   4. Extract zipped folder
   5. Open notepad or any text editor
   6. Go to the folder "install", right open the file "install2.bat" using notepad. If you do not see the file, select "All filetypes" from the dropdown in the windows explorer search, it defaults to only look for .txt files.
   7. Change the 'YOURPATHHERE' to the address of the folder that contains "install2.bat". For example if my folder with the file is C:/downloads/craniobot/install I would replace 'YOURPATHHERE' with 'C:/downloads/craniobot/install'.
   8. Save the file.
   9. Add python to your path variable in windows as described in this tutorial <https://www.pythoncentral.io/add-python-to-path-python-is-not-recognized-as-an-internal-or-external-command/>
   10. Right click the file "install2.bat" and click run as administrator. This will download the following python packages.
   11. python packages
       * numpy-1.14.2
       * matplotlib-2.2.0
       * plotly-2.5.0
       * pyserial-2.7
2. Upload arduino file (~3-10 minutes) See <https://www.arduino.cc/en/Guide/ArduinoDue> to familiarize youself with the arduino editor and the arduino due and for trouble shooting help.
   1. Connect the Arduino Due to the computer using a USB
   2. Double click the file "Surface\_Probe.ino" located in the craniobot download folder. This will open the arduino editor.
   3. In the arduino editor click Tools>>BOARD>>Board Manager
   4. Search "due" in the search bar
   5. Install the Arduino SAM Boards file
   6. In the arduino editor click Tools>>BOARD>>Arduino Due (Programming Port)
   7. In the arduino editor click Tools>>PORT>>COM1. Note: The COM port may be COM1, COM2, COM3, COM4, COM5,... depending on the number of USB ports using serial communication on your computer. To figure out which port is the arduino, take note of all COM ports and then unplug the arduino USB. The port that is the arduino will no longer show up.
   8. Click the check mark beneath the file button. This will verify the code.
   9. Click the arrow next to the check mark, this will upload the file to the arduino.
3. Change tinyG COM port in python code (~3-10 minutes)
   1. To identify the COM port of your tinyG, go to the arduino application. You can search for arduino in your computer to find the application.
      * plug in your tinyG USB connetion and make sure your tinyG is turned on.
      * In the arduino editor click Tools>>PORT and look at the COM options (i.e. COM1, COM4, COM5 is shown)
      * Unplug the tinyG USB from your computer
      * In the arduino editor click Tools>>PORT>> and look at the COM options.
      * The COM of your tinyG port will no longer be on the list (i.e. COM1, COM4 is shown, this would mean COM5 is the tinyG port in this example).
   2. Plug the tinyG USB back into the computer
   3. Go to the file in the folder python titled "CNCController.py".
   4. Right click the file and click edit with IDLE
   5. Change the COM4 in "self.port = 'COM4' " to the serial port of your tiny G controller (i.e. in our example COM5 is the tinyG port so we would write self.port = 'COM5')
   6. Save the file and close the editor.

**Demo and instructions for use**

1. Instructions to run
   1. If you are doing a surgery, prepare the mouse as you would for a craniotomy. If you are doing a demo, prepare a dummy (we use a 50mL falcon tube laid on its side with a post it on the surface) and position it where you would expect to position the mouse skull on a stereotax.
   2. First, we will define the points to use for the craniotomy. All reference coordinates are relative to bregma and are specified in the file “pointGen.py” inside of the python code folder in units of mm in the variable logo\_coordinates. To edit this file, right click the file and click edit with IDLE.
      1. Several example paths are commented out. For example this is a rectangle over the right hemisphere of the skull example.

#logo\_coordinates = [[1.1,2.6,2.6,1.1,1.1],

# [2.5,2.5,-4,-4,2.5]]

The first vector are the X coordinates ([1.1,2.6,2.6,1.1,1.1]) and the second vector are the Y coordinates for each corresponding X coordinate ([2.5,2.5,-4,-4,2.5]). To use this path, remove the “#” . Note: only one logo\_coordinate should be defined at a time (i.e. the result should have the “#” in front of them). Save this file and continue to the following steps.

* 1. To run all commands we will use the command line. Search for command prompt in the windows search bar, and right click run as administrator.
  2. Locate the path to the craniobot folder called python code. I may look something like C:\downloads\craniobot\python code
  3. In the command prompt type

cd C:\downloads\craniobot\python code

* 1. Open the python command line. In the command prompt type

python

* 1. Ensure the tinyG is not in an error/sleep state. Since this is the start of the procedure, reinitialize everything via switching power off and then on.
  2. In the command prompt type

tinyG\_startup.py

it should connect to COM port 4

* 1. In the case that it was not reinitialized, type

tinyG.wakeUp()

to wake it up, or a jog command input twice will wake it up, then jog in the desired step/rate.

* 1. Jog the end mill up (and away in xy directions if needed) to situate the mouse in the stereotax:

tinyG.jog(“z”,1,200)

input multiple times can accomplish this.

* 1. Remove the skin, fat, and fascia covering the dorsal skull.
  2. Move the probe away from the mouse using jog commands, usually 15mm right and 15mm down. In the command prompt type

tinyG.jog(“x”,15,200)

then tinyG.jog(“y”,-15,200)

and put the probe onto the mouse.

* 1. Move the probe above bregma using jog commands. In the command prompt type

tinyG.runSingleProbe()

The probe moves down to bregma and stops when contact sensor switch opens

* 1. In the command prompt type

tinyG.setOrigin()

* 1. For non-cicular craniotomies, make sure logo\_coordinates contains two vectors for x and y coordinates in mm. Use BrainWindow(step\_size) to generate pilot points. Ex:In the command prompt type

bw = BrainWindow(0.3)

would be logo\_coordinates with the minimum step size between points being 0.3 mm; extra points would be added to ensure this.

* 1. In the command prompt type

tinyG.currentPosition()

to check if origin registered correctly, and at origin

* 1. In the command prompt type

tinyG.runProbe(bw.gCode)

to begin probing at each point.

* 1. Store tinyG.probe\_output data into external container, ex: In the command prompt type

bw\_out = tinyG.probe\_output

* 1. Move probe away from skull, switch end mill to cutting tool, move the tool back to bregma (very precisely), all using jog commands
  2. In the command prompt type

tinyG.setOrigin()

* 1. Generate mill path:

path = MillPath(tinyG.probe\_output, depth)

If you are doing a skull thinning procedure, comment out lines that return the milling path to the initial point before generating the mill path in generate\_milling\_commands (lines 49 to 58). Usually I name my milling path by putting the depth into the variable, e.g. w50 = MillPath(bw\_out, 0.050) would be for milling 50 um deep. Generate as many milling paths as you anticipate needing (you can generate them in later/earlier steps).

* 1. Examine the mill path to make sure there aren’t false positives.
  2. In the command prompt type

path = tinyG.currentPotision()

* 1. In the command prompt type

tinyG.runMill(path.gCode)

or going along with previous examples, tinyG.runMill(w50.gCode)

* 1. repeat for as many iterations/paths as needed.
  2. Finished with craniobot part of surgery. You can now remove the skull.

1. Expected output
   * The expected outcome is a craniotomy in the shape specified in the milling path coordinates.
2. Expected run time is provided in the main manuscript.