

## How to set up the Raspberry Pi and Gertbot as a tomography controller

- written by Burkhard Schillinger, Heinz Maier-Leibnitz Center, Technische Universität München, Germany
- Tested and commented for Raspberry Pi 3 by Aaron Craft, Idaho National Lab, USA
- [\[Comments by Aaron Craft in BLUE\]](#)
- Buy a Raspberry Pi (model B, B+ or 2) , a Gertbot (NOT gertboard!), a power supply, a 8 MB micro SD card and maybe a HDMI-to VGA converter (if you want to use an old monitor) from [www.farnell.com](http://www.farnell.com). There are lots of other suppliers for the Raspberry Pi, the Gertbot can be directly ordered on [www.gertbot.com](http://www.gertbot.com).  
The Raspberry Pi Model 3 would also work, but the model 3 is the most powerful of the series and requires an extra non-standard power supply that delivers 5.25 V instead of 5 V. That can be bought at Farnell as well, but it removes the flexibility to use standard components or power supplies later on, and that special requirement will soon be forgotten when someone else wants to work on it. The Model 3 has some differences compared to the others that were tested and commented by Aaron Craft.

I purchased a Raspberry Pi3 for this project. There are many other differences with the Pi3 that require different setup than that described in this guide for the Pi2. I don't regret getting the Pi3 though, as working through some of the issues with the Pi3 may benefit others using this guide in the future.

I purchased a CanaKit Raspberry Pi 3 Ultimate Starter Kit for \$90 from Amazon.com, which included a Pi3, clear case, which included the power supply, a 32 GB microSD chip, USB/micro-SD adapter, wiring breadboard and breakout ribbon cable, and an assortment of resistors, diodes, and wires. The microSD card is pre-loaded with NOOBS.

- You may ask the author [Burkhard.Schillinger@frm2.tum.de](mailto:Burkhard.Schillinger@frm2.tum.de) for a fully working image of the operating system and control program, or do the following steps yourself.

Setting up the operating system was relatively straightforward for the Pi3. Upon startup, the system allows you to connect to a wireless network. Once connected, NOOBS allows the user to select and download a number of operating systems. Raspbian Jesse was downloaded and installed. Using NOOBS, this system did not require manually imaging the microSD card.

Additionally, reformatted the microSD card and manually imaged Raspbian Jesse to the microSD card per the instructions in this guide. This worked as well.

- Copy the operating system 'Jessie' to your SD card, as described on [www.raspbian.org](http://www.raspbian.org) or many other web sites. Insert the SD card into your Raspberry, then connect keyboard, mouse, network, monitor (if you have an old one, use a HDMI to VGA converter cable) and power.
- Go to [Menu/Preferences/Raspberry Pi control \(Menu/Preferences/Raspberry Pi Configuration\)](#) and expand your file system, then set your language and locale as appropriate. The option to "expand your file system" was grayed out and not functional on my system, so I continued without that step. There may have been an issue with permissions.

- 
- You MAY open a console and update your system by typing 'sudo apt-get update' and 'sudo apt-get upgrade', but it is not required for the controller. If you get error messages in the update process (as was common at the time of writing, when 'Jessie' was still quite new), copy the image from your PC to the SD card again, and do NOT do the update.
- Now the internal serial console must be disabled to free the serial port for the Gertbot. The following text was copied from [http://www.robot-electronics.co.uk/files/rpi\\_serial\\_setup.doc](http://www.robot-electronics.co.uk/files/rpi_serial_setup.doc) :
- (Enabling UART is **VERY** different for a Pi3. This is described separately after the following section.)

---

### **Enabling the UART port on a Raspberry PI**

By default the serial port on the Raspberry PI is configured for console input/output. If you wish to use the serial port in your own software then you must make the following changes. First use nano to open /boot/cmdline.txt and edit it to disable boot info being sent to the port.

```
sudo nano /boot/cmdline.txt
```

The contents of the file should look a bit like this:

```
dwc_otg.lpm_enable=0 console=ttYAMA0,115200 kgdboc=ttYAMA0,115200
console=ttY1 root=/dev/mmcblk0p2 rootfstype=ext4 elevator=deadline
rootwait
```

Remove the text highlighted in red, this will stop boot info being sent on startup. Press Ctrl+O to write the changes and Ctrl+X to exit nano.

Now we need to edit /etc/inittab to disable console login on the serial port. Open this file with nano

```
sudo nano /etc/inittab
```

Look for this line:

```
T0:23:respawn:/sbin/getty -L ttYAMA0 115200 vt100
```

And comment it out by adding # to the start of it. You can then save this file and exit nano like you did before.

Now you have made these changes we can reboot the Raspberry PI.

```
$sudo reboot
```

Your serial port should now be ready for you to use in your own software.

---

### **Enabling the UART port on a Raspberry PI 3**

By default the serial port on the Raspberry PI is configured for console input/output. If you wish to use the serial port in your own software then you must make the following changes.

First use nano to open /boot/cmdline.txt and edit it to disable boot info being sent to the port.

```
sudo nano /boot/cmdline.txt
```

The contents of the file [on the Pi3](#) should look a bit like this:

The Pi3 `cmdline.txt` file is slightly different:

```
dwc.otg.lpm_enable=0 console=serial0,115200 console=ttyl  
root=/dev/mmcblk0p7 rootfstype=ext4 elevator=deadline fsck.repair=yes  
rootwait
```

Remove the text highlighted in red, this will stop boot info being sent on startup. Press Ctrl+O to write the changes and Ctrl+X to exit nano.

The resulting line looked like the following:

```
dwc.otg.lpm_enable=0 console=ttyl root=/dev/mmcblk0p7 rootfstype=ext4  
elevator=deadline fsck.repair=yes rootwait
```

The file `initab` does not exist on the Pi3.

To enable UART, do one of the following, then reboot the system.

1. Open `Menu/PreferencesRaspberry Pi Configuration`. In the `Interfaces` tab, enable `Serial`.
2. Edit `/boot/ config.txt` [`sudo nano /boot/config.txt`] to include `"enable_uart=1"`.

Now you have made these changes we can reboot the Raspberry PI.

```
$sudo reboot
```

Your serial port should now be ready for you to use in your own software.

---

---

- 
- 
- Go to [www.gertbot.com](http://www.gertbot.com) using the browser on the Raspberry Pi and download 'Gertbot Debug GUI (.tgz)' from the Executables section. The Browser will offer to unpack it – place it in a new directory 'gertbot' or similar in the HOME directory. Then download 'Python-drivers (.tgz)' from the drivers section, and place the unpacked file in the same directory. Also download all manuals from the manuals section.
  - Power down the Raspberry Pi, and connect the Gertbot. It fits only one way (Fig.1).

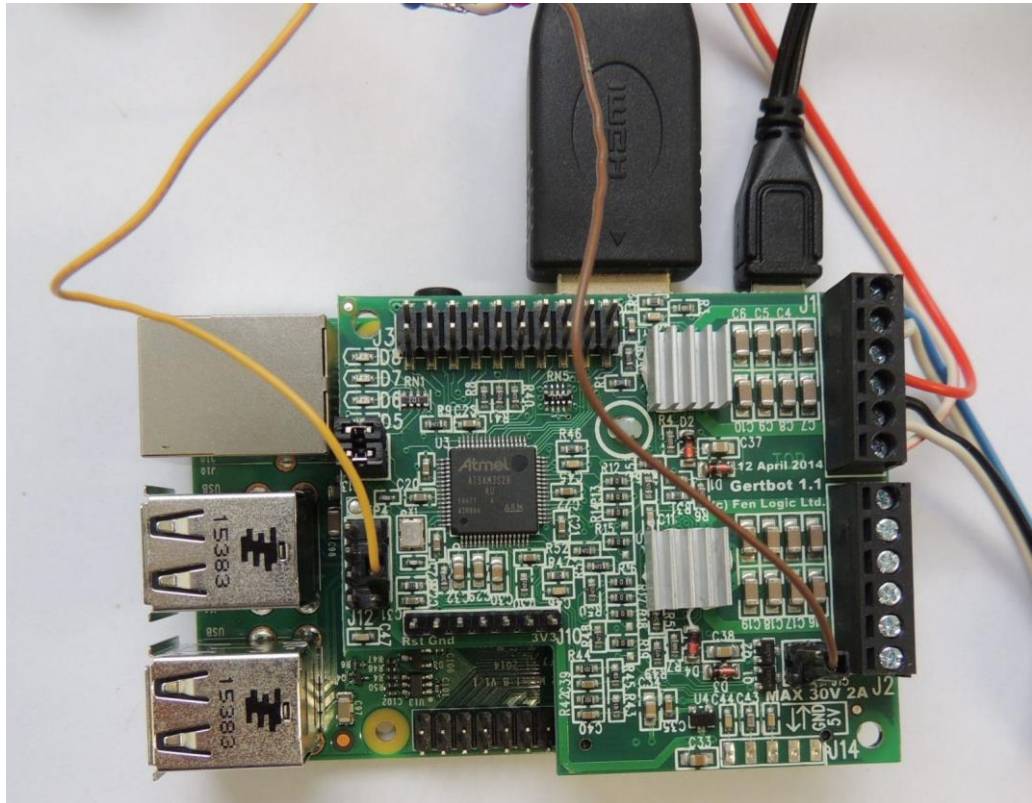


Fig.1: Raspberry Pi with mounted Gertbot. The brown cable on the right connects to the open collector driver ODO, the yellow cable on the left to +5V on the internal serial port.



Fig.1b: Raspberry Pi 3 with mounted Gertbot. The yellow cable on the right connects to the open collector driver ODO, the brown cable on the left to +5V on the internal serial port.

- After boot, you can call the test software GUI in a shell window with 'python3 ./gertbot.py'. Caution: 'python3' MUST be used, python or python2 will not work.  
I used the executable to start the GUI using the command `./gertbot` from the directory containing the executable. This opens the GUI for further testing.

If you open the gertbot GUI and try to connect, the log window opens and the program freezes after displaying "Board 0...". After much toiling to figure out what was causing this error, including starting over completely a couple times, I eventually found a forum that had a simple but strange solution: disable the Bluetooth. It seemed strange at first, but it worked! Apparently, Pi3 uses UART for Bluetooth by default, while most other boards use GPIO for Bluetooth.

To disable Bluetooth:

Edit `/boot/config.txt` [`sudo nano /boot/config.txt`] to include, "`dtoverlay=pi3-disable-bt`".

Now the GUI should connect to the gertbot and find the board. To test whether things are working properly, open the GUI and try controlling the stepper motor.

- For further hardware installation, power the Raspberry Pi down.
- Open the mouse on your camera control PC, and solder two wires to the connections of your left mouse button. Optionally, install a socket for a phone/stereo jack cable on the side of your mouse, and connect the wires to it (Fig.2).



Fig.2: Connecting the extra wires and connector to the left mouse button.

- Connect your stepper motor to the screw terminal 1 of the Gertbot according to the manual. Caution: Many motors with a ribbon cable are not wired coil1-A, coil1-B, coil2-A, coil2-B, but instead coil1-A, coil2-A, coil1-B, coil2-B. If your motor only vibrates with the test program instead of turning, you got it wrong. Check your motor's data sheet in the internet. The motor must to be powerful enough to drive the rotation table without slipping.
- Connect an external power supply (typically 12 V) for the stepper motors to the screw terminal according to the manual.
- Connect a 5V reed relay with its ground pin to the first of the open collector outputs (OD0). This is the lower left pin of the 4-pin connector next to the bottom of the screw terminal. The

relay contains a diode in parallel to its coil, so polarity does matter. Connect the Vcc side of the relay coil to 5V on pin 3 of the J12 connector. Connect the switch contact of the relay to another telephone jack socket, and use a double phone/stereo jack cable to connect it to your mouse. Fig.3 shows an overview of all components.

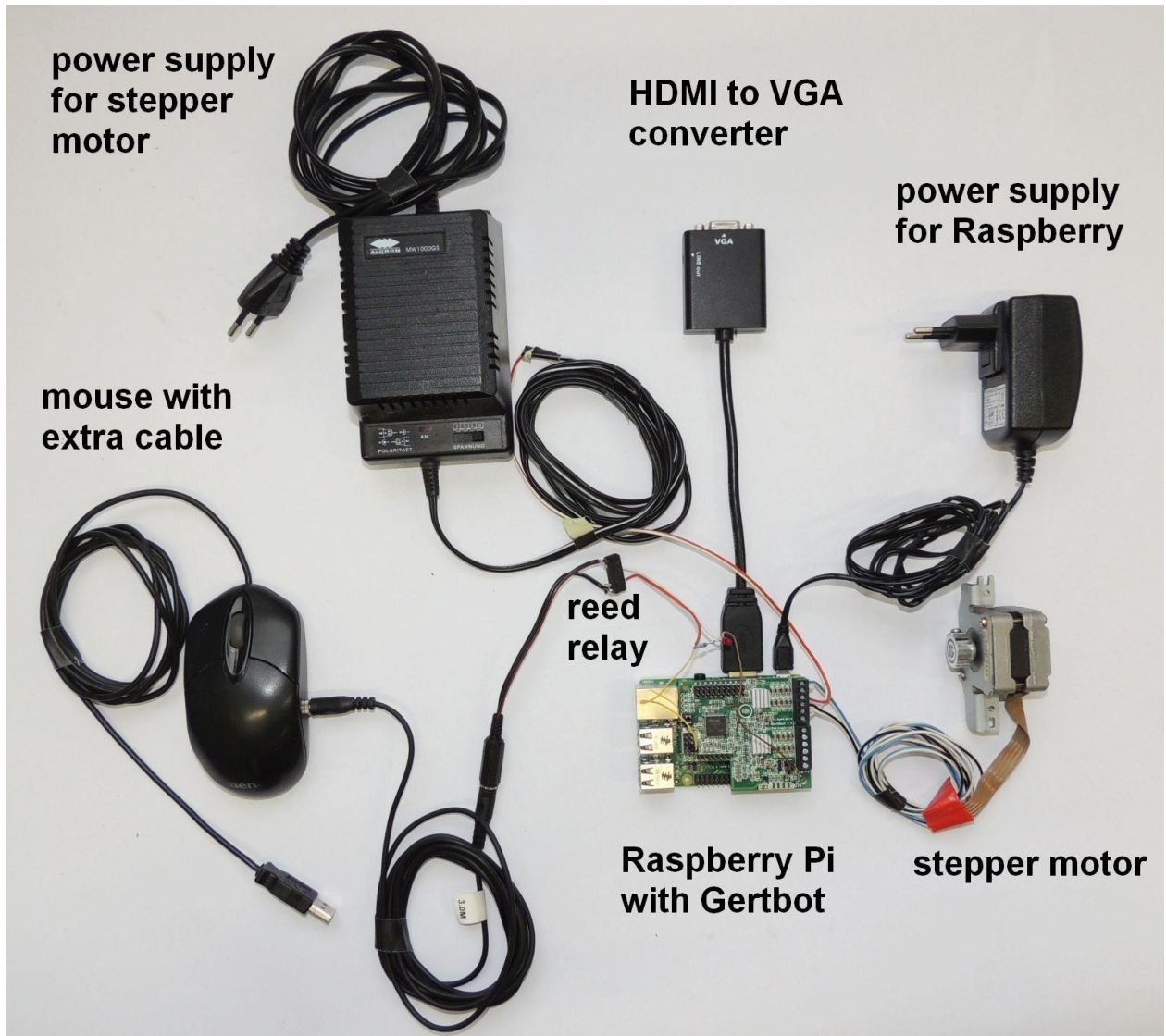


Fig.3: Overview of all components for the controller.

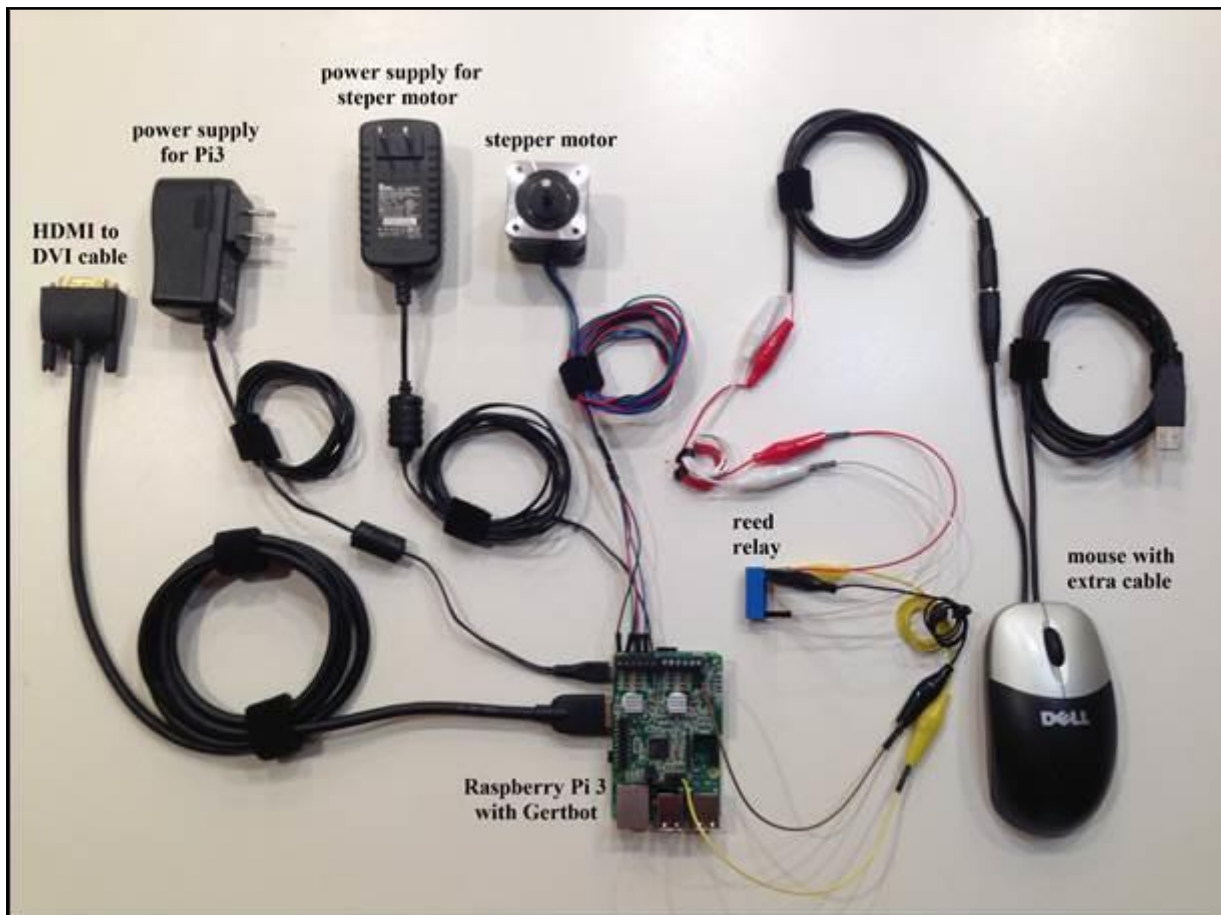


Fig.3a: Overview of all components for the controller.

### Current Status:

The Pi3 is set up and communicating with the Gertbot. The Gertbot GUI can communicate with the stepper motor and move the motor. However, the motor may not be connected properly, even though I have tried all combinations of connections for the coils. I may try a different stepper motor.

I have yet to get the script working on the controller.

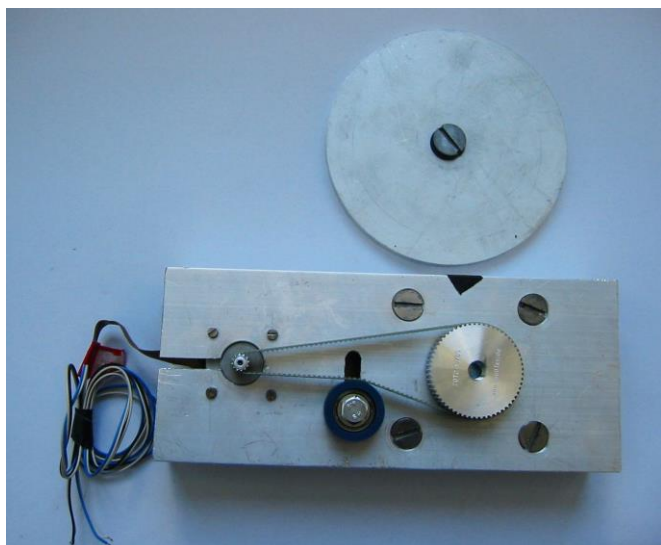
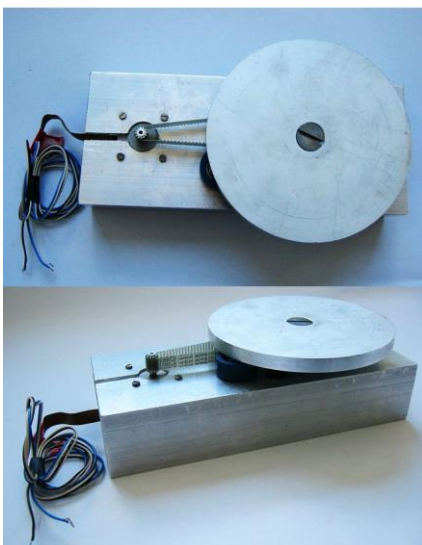
- Copy the tomography control script at the end of this text into a text file called 'tomo\_test01.py'. The easiest way to copy it to the Raspberry is to send it to yourself by Email, then call your Email on the Raspberry Browser, and copy it to the gertbot directory.
- Boot your camera PC and camera, then record an image series of several images, if possible with autosave, with 1 msec exposure. Note the time until the next image is started. This is the turnaround or cycle time of the camera.
- Edit the file, and set the parameter for 'reduction' to the reduction of your rotation table. It should take  $200 \times \text{reduction}$  steps for one full rotation. Then set the cycle time to the value you measured above, plus 0.3 seconds or so for safety.

- Define/edit the user parameters for rotation range of 180/360 degree, exposure time and number of images. Save the file.
- Initiate an image sequence with individual trigger on your camera PC, then place the mouse pointer on the start button for the individual images.
- Go to the Raspberry, and start the tomography script with 'python3 tomo\_test01.py'.
- Once the whole setup works, the extra monitor, keyboard and mouse for the Raspberry Pi are no longer required, since it can also be accessed via a SSH shell through the network running on the camera PC. In that case, a five second wait must be included/uncommented in the control script to allow the operator the time to move the focus of the mouse from the SSH window to the camera software and its start button.

---

## Optionals

- The second open collector output may be used to drive the beam shutter. Add waiting times accordingly.
- The second motor output can be used to drive a translation table that carries the rotation table, and moves it in and out of the beam. Together with the shutter driver, you can also automate the recording of open beam and dark images. It is essential to connect limit switches for the translation table, and read them according to the Gertbot manual.
- To obtain quick results, a rudimentary rotation table was built with the stepper motor shown above, a toothed belt and two cogwheels with a 5:1 reduction ratio, as shown in Fig. 4. This allows for 1000 steps per full rotation, sufficient for a good quality tomography. The belt tensioner is simply made from a ball bearing. The bearing for the rotation table can support 10 kg. However, it must be stressed again that this is a quick-and-dirty solution, with no adequate precision whatsoever. Its sole purpose is to obtain first results that may support a grant application for funding of advanced components like a precision rotation table.





- Fig.4: Rudimentary rotation table with 5:1 reduction ratio built with a stepper motor, a toothed belt and two cogwheels.

===== Control script tomo\_test01.py =====

#

# Test program for tomography sequence

#

# the Gertbot drivers

import gertbot as gb

import time

# This is for the board and motor channels:

BOARD = 0 # Which board we talk to

ROT = 0 # channel for rotation table

TRANS = 2 # channel for translation table, yet unused in this script

STEPS=100

STOP = gb.MOVE\_STOP

#parameters for motors and tomography

reduction = 5 # gearbox reduction of rotation table,

```

# stepper motor alone does 200 steps/rotation
angleconv = reduction * 200/360 # number of steps per degree, float value!

# ===== User parameters

n_angles = 401 # number of angles for tomography
ExpTime = 1 # Exposure time for camera
cyctime = 1 # Cycle time for camera without exposure time
rotrange = 360 # 360 for full rotation, 180 for half rotation

# uncomment the lines below down to the sleep command if you are running this script
# from a SSH shell on the camera PC.

# print("")
# print('Now move your mouse from the SSH shell window to the camera window, ')
# print('click into the window once to activate it, ')
# print('and move the mouse pointer on the start button for your image sequence. ')
# print('This script will begin execution in five seconds. ')
# print("")
# time.sleep(5)

# =====

#

# gb.move_stepper(board,channel,steps)

#      ^^^^^ ^^^^^^^ ^^^^^^^^^
#      ||||| ||||| ||||| |||||

```

```

#      ||||| ||||| ||||| |||||
#      ||||| ||||| ||||| ++++++-----<steps
#      ||||| ++++++-----< Which motor on board 0,1,2,3
#      ++++++-----< which board 0,1,2,3

# =====

# Main program

# Open serial port to talk to Gertbot
gb.open_uart(0)

# Setup the channels for stepper motors
gb.set_mode(BOARD,ROT,gb.MODE_STEPG_OFF)
gb.set_mode(BOARD,TRANS,gb.MODE_STEPG_OFF)

# =====

paststeps=0

for i in range (0,n_angles+1):
    angle = float(i)*rotrange/n_angles
    print('angle= ', angle)
    STEPS=int(angle*angleconv) # total number of steps from zero position
    STEPS= STEPS-paststeps # number of steps from previous position
    paststeps=paststeps+STEPS # total number of steps to current position
        # this is required to eliminate rounding errors
    print('paststeps= ', paststeps)

```

```
gb.move_stepper(BOARD,0,STEPS) # test
time.sleep(1+STEPS*.01) # default is 100 Hz
gb.activate_opendrain(BOARD,1,0) # camera pulse on: click mouse button
    # via open collector and relay
time.sleep(.2)
gb.activate_opendrain(BOARD,0,0) # camera pulse off
time.sleep(.3)
time.sleep(cyctime+ExpTime) # wait for camera to do exposure

print("")
print('tomography complete!')
print("")

# on exit stop everything
gb.emergency_stop()
```