Software notes:

AMiGUI.py program:

The AMiGUI.py program is included as a separate file. The program is written in Python3 and uses Tk to create the graphical user interface. The program will stop with an error if it does not find a control file named AMi.config and a directory or symbolic link named "images" in the directory from which the program is launched. To launch the code type `python3 AMiGUI.py`.

Sample AMi.config file:

```
12     8     1     # number of positions on x and y, then samples at each position
  140.776   26.256   15.856  # coordinates of the top left sample
   41.760   25.040   15.213  # coordinates of the top right sample
  140.008   89.497   17.232  # coordinates of the bottom left sample
   41.277   88.138   15.729  # coordinates of the bottom right sample
   0.0000   0.0000  # fractional offsets of sub-sample
    0.100 # zstep - the spacing in z between images
     3     # nimages - the number of images of each sample
lysozyme   # sample name
mcsg2_ab2     # plate_name
```

Please note that each additional sub-sample will cause one additional fractional offset line to be added to the file. The values in the file must be separated by at least one space. Also it is best to have the plate name be unique.

Software installation

If you did not purchase a microSD card with the raspian operating system pre-loaded, the system can be downloaded from https://www.raspberrypi.org/downloads/. You will need the system with the desktop included. Once the microSD card has been installed in the raspberry pi and the pi has been powered up, copy the software (AMiGUI.py) and the configuration file (AMi.config) to the pi. This can be done via web browser or via a USB thumb drive.

When you start the system for the first time, your display will most likely have a black border around the edge. To use the entire display, edit `/boot/config.txt` and uncomment the line that says

`disable_overscan=1`. Also, go to Preferences > Raspberry Pi configuration > Interfaces and click the circles to enable SSH and the Camera.

Images directory:

As noted in the main text, image files will be written to a directory named "images." This should reside in the same directory that contains the AMiGUI.py program. There are at least three options for storing the images.

1) The images can be stored on the SD card on the pi. Micro SD cards are less reliable than hard disks, particularly when subjected to extensive read/write cycles. Thus, saving images to the SD card is okay for testing, but not recommended for routine use.

To have the images written to the SD card, simply create a directory on the pi named images (`mkdir images`). In this case you will most likely use sftp to move the images to another location after they are collected.

2) The images can be stored on an external disk or thumb drive connected to the pi via USB. This avoids writing to the SD card and allows the images to be transferred manually to another computer. Note that you must format the stick using the ext4 (linux) file system for this to work.

Connect the external drive or USB stick - it should mount automatically. To find its location use the command df (`df`). You should see an entry for the external drive (i.e. /media/pi/USB_STICK). Now create a symbolic link named images that points to the external USB drive (i.e. `ln -s /media/pi/USB_STICK images`).

3) The images can be stored on a disk on another computer that is shared via NFS. This is the most convenient arrangement, but also a bit more complicated to set up. The directions below assume you have connected the Raspberry Pi via ethernet to a local network and that the computer with the external disk is running Linux. In the example, the Raspberry Pi has ip number 192.168.2.8 and the computer with the hard drive has ip number 192.168.2.18. This is just one of many possible configurations

involving a shared disk. (Note that NFS can cause security issues if implemented on an unprotected network. To help mitigate this risk, users may choose to create a special user for the images.)

First export a directory on the disk you wish to share by adding a line like the one below to **/etc/exports** on the computer with the hard disk you wish to share:

```
/home/microscope/images 192.168.2.8 (rw,sync,no_root_squash)
```

Now issue the following commands on the computer with the hard disk:

```
sudo systemctl start nfs-kernel-server.service
sudo chmod 777 /home/microscope/images
sudo exportfs -a
```

Now set a static IP address on the rasperry pi by editing the file **/etc/dhcpcd.conf** and adding lines that define the interface, static IP address, router, and domain name server. For example:

```
interface eth0
static IP_address=192.168.2.8
static routers=192.168.2.1
noipv6
static domain_name_servers=130.64.215.166
```

Then create a symbolic link named images that points to the external disk.

```
sudo mount 192.168.2.18:/home/microscope/images /home/pi/images
```

If the Pi is turned off, and back on, you will need to issue the last line again to reestablish the connection to the remote disk. Adding this line to the end of **`/etc/bash.bashrc`** will cause it to be executed automatically whenever a terminal window is opened. Although it is technically the correct thing to do, it is best not to add the remote disk to the /etc/inittab because the Raspberry Pi will not boot if the remote disk is not found, and you will be locked out.

Directory structure:

AMiGUI will create and update a directory structure that is designed to help keep the thousands of images it collects organized by project, plate name, and date. The contents of this directory structure are summarized below, where tabs denote subdirectories.

`images` directory

sample directory (i.e. **`lysozyme`**)

plate directory (i.e. **mcsg1_ab3**) the name should be unique

snaps directory (only created if there are manually takent snapshots)

individual snapshots (i.e. **D6a_Mar-19-2019_02:1:32PM.jpg**)

process_snap.com (only written if a z-stack of a single well is taken. This is done by right-clicking the snap button.)

date directory (i.e. **Mar-22-2019_11:25AM**) autoimaging output images

rawimages directory

individual images (i.e. **A1a_0.jpg, A1a_1.jpg, A1a_2.jpg…**)

copy of the configuration file used for autoindexing (i.e. **AMi.config**)

script file for processing the images (i.e. **process_mcsg1_ab3.com**)

processed images (i.e. **A1a.tiff, A1b.tiff, A2a.tiff…**)

these are the final, digitally depth of field enhanced images.

Grbl setup:

The Arduino board uses the program Grbl to control the three stepper motors and respond to the limit switches. Before running AMiGUI, you need to configure Grbl using a text-based terminal. The arduino will remember the altered settings, so this only needs to be done once.

With the Arduino board connected to the Raspberry Pi, install picocom (`apt-get install picocom`). Then issue the following commands:

`picocom /dev/ttyUSB0 -b 115200 -l` (connects to the Arduino board)

`$21=1` (activates hard limits so the machine stops when switches are contacted)

`$22=1` (enables homing so the machine can find its zero position)g

`$3=7` (changes default direction, so + translations move towards the right, back, and top)

`$23=7` (sets zero at the left, front, bottom position)

`$$` (show current settings)

`$H` (test homing cylce – stage should move to the front left bottom position)

The `$H` command above is a test of the settings and switches. When you are done, just close the terminal window.