Hacking a microscope @OpenFIESTA

xiǎn wēi jìng

显微镜

version 1.0

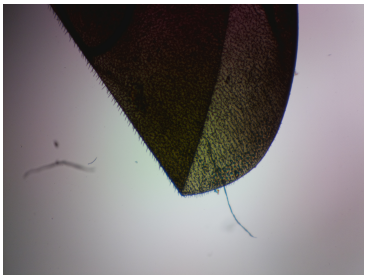
Step by step **guide**

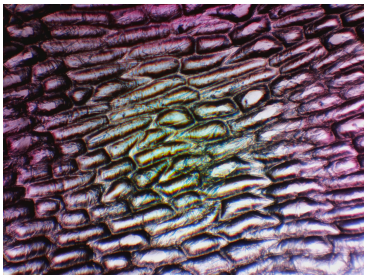
zhú **逐**

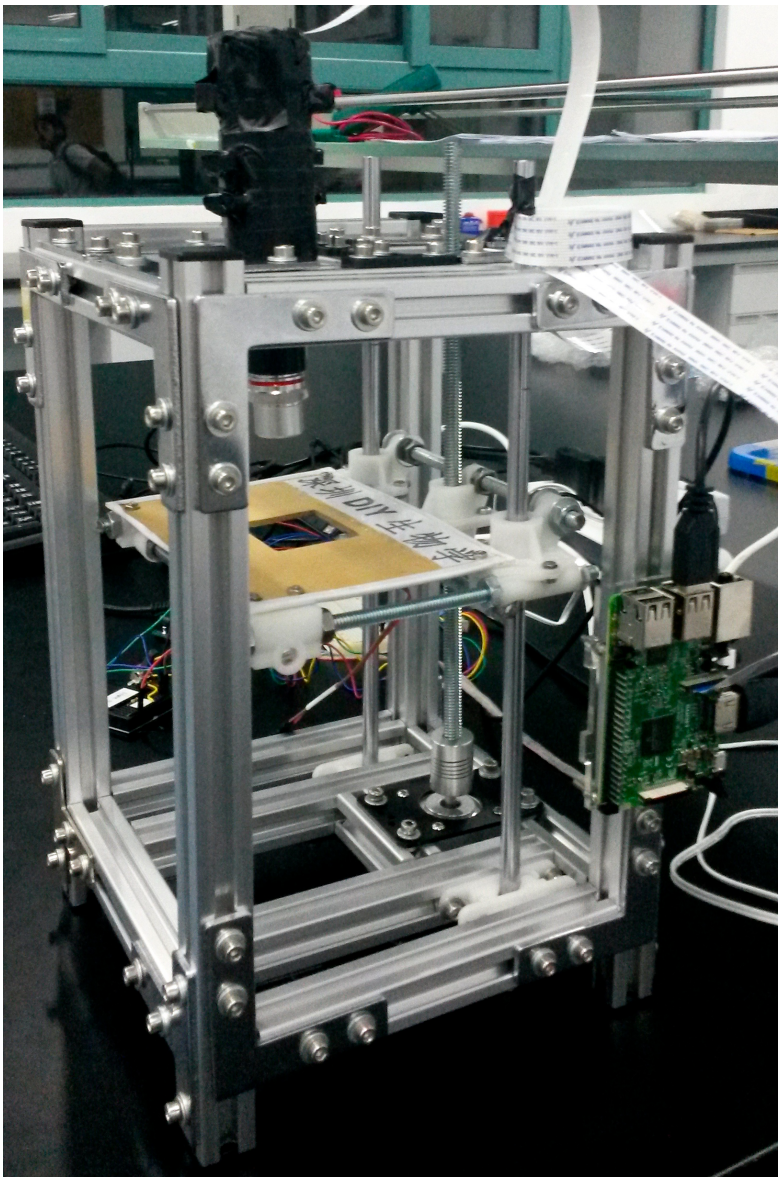
bù **步**

zhǐ **指**

nán **南**







About the cover

The OpenFiesta microscope version 1.0, (bottom panel). The wing of a wasp species which flew into the lab (top left panel) in Shenzhen seen at 4x and pictured by a Raspberry Pi camera using the open source application *raspistill* running on a Raspberry Pi computer. The top right panel shows the skin of red onion from the campus canteen captured with such a same set-up.

Hacking a microscope with friends in 深圳 after thyphoon Hato, 2017 年 8 月 26 日

The following guide documents the hacking done while building a V-slot based open source microscope. As a starting point in hardware space, we considered the OpenBeams based microscope from the OpenLabTools. This guide is to document what we have learned hacking together with students of OpenFIESTA while enjoying the power of thyphoon Hato thinking on global sustainability goals at Tsinghua University Graduate School in Shenzhen. We are infinitely indebted to all the people and all the forces which made our visit to OpenFIESTA possible. Also we are infinitely thankful to our friends Luping Xu (徐芦平), and Ji Li (李季), as well as to all their collaborators and specially to the talented students who's enthusiasm and hard work had made this adventure an intellectual pleasure. We are specially thankful to Tobey (黄园芳), Aileen (廖玲), 刘晨光, 毛运浩, 段桂春, 吴越 and Alfredo L'Homme who stayed with us teaching and learning late at night. To all of you, 多很谢谢!

J&J
深圳 中国
2017 年 8 月 30 日

1.-Introduction

During a very hot summer in Guangdong (广东) province, we were thinking about sustainability while enjoying the powerful thyphoon Hato and the intellectual challenges emerging in our minds while in the **2017 i-SDG Shenzhen assembly**. Together with students of the IID and BIO[3] master programs of OpenFIESTA, we were wondering about *quality education,* and *industry innovation and infrastructure*. At Shenzhen speed, with the help of 李季, we started building a microscope using hardware available from the local markets (华强北, 淘宝网, among others). Thus, we set to explore the Shenzhen innovation ecosystem (深圳创新生态系统) and use it to ***learn by doing*** while mixing hardware machines (五金机) with bio (生物). Together, we all learned and speculated about mixing these two disciplines (学科) while shaping the open science hardware movement.

The starting point of our adventure was the **OpenLabTools Microscope**[1] from Cambridge, 英国. Their OpenBeams based frame (*see* Fig. 1) and over all construction was the basis of our approach. We used the OpenLabTools documentation (available online) and recycled it extensively in this guide. We are extremely thankful to the authors of the OpenLabTools microscope documentation for their contribution to our adventure, by making the road we travel a lot easier. In the inner cover of this guide you can appreciate how printouts of the original documentation (which was translated to 普通话 by the students) displayed in a board over the workbench acted as a road map. This new guide is our attempt to document last summer's adventure, hoping it will help students to take their project to the next level and help us all to soon see the 显微镜 version 2.0.

This "Shenzhen version" of the OpenLabTools microscope (*see* image on the cover) is tailored at the local supply of components from the Shenzhen ecosystem. Shenzhen, being the world´s major electronics and hardware components manufacturing hub, provided us with an unlimited supply of possible parts to *evolve hardware* and we hope to soon leverage this again to further evolve robotic image capturing systems sampling spatially distributed cell communities.

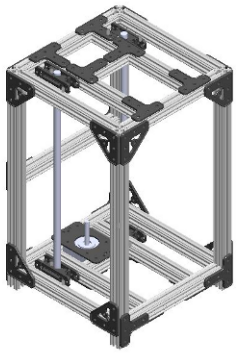


Figure 1. OpenLabTools microscope frame was designed at Cambridge, UK and. it is based on *OpenBeams* (*Image from OpenLabsTool's original guide*)

One particularity of the Shenzhen's supply ecosystem is that you can find both original as well as look-alike ("shanzhai", 山寨) versions of almost every hardware part available. The advantage of this, is that you can find very cheap parts to build and test some version of your prototype meeting whatever constraints you might have in time and/or cost. The drawback that comes with it, is that it is hard to know if you are buying good quality parts. The benefit however is that you can prototype at Shenzhen speed and move fast into the next developmental cycle.

[1] http://openlabtools.eng.cam.ac.uk/Instruments/Microscope/

2.-The V-slot structure

The 3D structure materializing the microscope was constructed using open source 20x20 (mm, cross section) **V-slots**[2] aluminum extrusion beams (*see* Fig. 2). The microscope is made of three basic mechanical components: (1) The frame, (2) the stage, and (3) the linear translation axis.

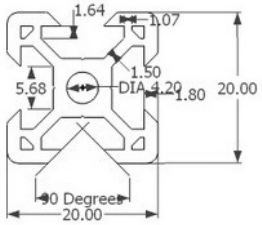


Figure 2. 20x20 V-slot beam profile. we used three different lengths of 20x20 mm profile V-slot beams

2.1.- Frame

We aimed to implement a fixed point in space to place a 3D printed optical clamp holding the optical imaging system and a linear translation axis housing a very simple microscope stage which can be moved *up* and *down* (Z-*axis* translation of the microscope stage) to allow an image coming from the sample (via transillumination by a single LED located under the stage) to focus on the XY *plane* where the Raspberry Pi camera board has the light detector and which is parallel to the microscope stage plane. The frame's *hight* is defined by ***four beams*** of length 300mm.

Two **2D top and bottom** square **bases** (*see* Fig. 3) are attached together by these four 300mm beams forming a 3D structure acting as microscope frame. Each of these two square bases consist of *six beams* of length 150mm plus one beam of length 60mm. These beams are arranged in a **4+2+1** configuration. This way **four** 150mm beams form a square which has **two** central 150mm beams inside for alignment of imaging, magnification, and illumination systems. These two inner beams are separated by **one** small *single beam* (60mm length). The top square base hold the optical system. This is done by attaching a 3D printed optical clamp to the one small beam on the base, The bottom base is made to support the actuating power of the microscope and therefore we attached an OpenBeam stepper motor bracket to the base. These two square bases are aligned respect to one another by the four 300mm beams holding the frame structure together. This way the light that comes from the LED traverse the sample and enters the optical tube at the objective.

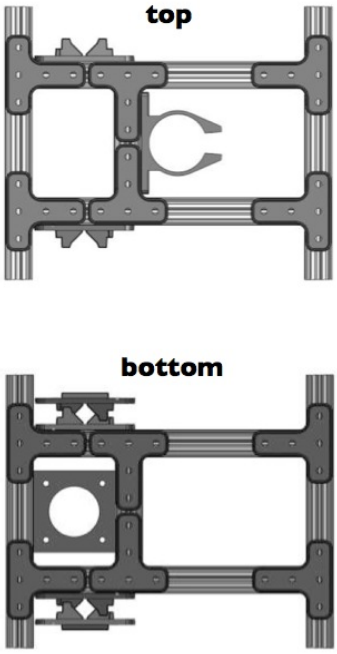


Figure 3. Top and bottom 2D bases. The top base is meant to old optics and the bottom to hold the motor needed to actuate the linear Z-axis used to focus stage and optics (*Image from OpenLabsTool's original guide*)

To put all the beams together into the final frame we needed V-slot compatible nuts and bolts as well as T-brackets and L-brackets. We ordered these from 淘宝网 and got them at OpenFIESTA in 24 hours. We also used 6 OpenBeam T-brackets when the V-slot T-brackets we ordered were too big to fit in the frame dimensions. On

[2] http://openbuildspartstore.com/v-slot-linear-rail/

top of these V-slot T and L brackets used, the bottom base needs four 3D printed OpenBeam T brackets holding the small beam together while the top base needs only two 3D printed OpenBeam T-bracket. We used metal OpenBeam T brackets in the bottom base to hold the small beam and in that way get a better attachment to the stepper motor. Again we had to enlarge the holes of the bracket.

2.2.-Stage

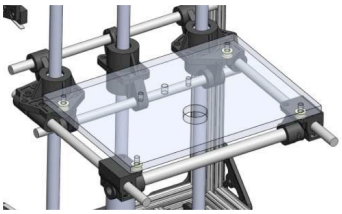


Figure 4. Microscope stage. The stage is made of M6 rods, 3D printed components and an acrylic or cardboard surface (*Image from OpenLabsTool's original guide*).

The stage, shown in Figure 4, is made of a frame consisting of five M6 threaded rods of 150mm length, M6 nuts and washers, and several 3D printed parts holding it together and attaching it to the linear translation axis. For the version 1.0 of the Shenzhen microscope stage we hand cut a piece cardboard and attached it to the stage frame using small screws and nuts available. The cardboard was attached to the four corners of the frame by screwing it to four of the 3D printed parts making up the corners of the stage. The files for all these parts are all available at http://shenzhen.keymer.cl and also at OpenLabTools repository. For more details see the OpenFIESTA microscope construction manual. There, They are listed using a number referencing them to these files. Use these when printing a new instance of the OpenFIESTA microscope to be sure. Also, notice that the Upper Left Mount piece in the original repository is corrupted and therefore use the file provided in the website. A hole in the middle of the cardboard is cut off to allow light coming form the LED illumination system traverse a microscope slide holding the biological sample being imaged.

2.3.- Linear translation axis

The linear translation axis is the mechanical structure that allows us to focus the microscope by translating the stage up and down along the *Z-axis* of the microscope (*see* Fig. 5). The linear translation is needed to map the *step-by-step* rotational motion of the stepper motor into movement along a linear axis. This is implemented in the microscope via a flexible shaft coupler coupling the leading M8 screw rod of 300mm length acting as main axis with a stepper motor resting on a motor bracket at the back and center of the bottom basal 2D plate of the frame. The rotation of the stepper moves the stage up and down the *Z axis* of the frame. The stage is also guided along the *Z-axis* by linear bearings hosted within the custom 3D printed parts attaching the two back corners of the stage to two smooth rods attached to the frame via 3D printed shaft clamps to each of the outer sides of the inner 150mm beams of the 2D bases at the top and bottom of the frame. The translation is done by two M8 nuts embedded into the two 3D printed parts making up the attachment between the stage and the

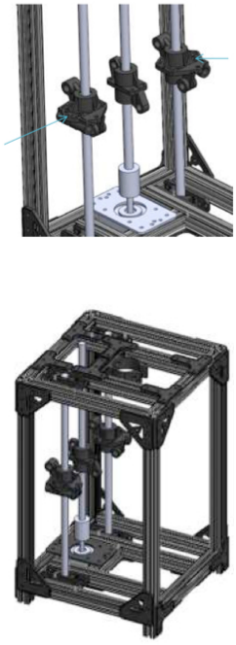


Figure 5. Linear translation. Movement up and down a *Z-axis* is powered by a 200 steps *Nema*17 stepper motor (*Image from OpenLabsTool's original guide*).

leading M8 rod screw.

3.-Optics

The optical system of the microscope consist of three main components: (i) objective, (ii) bi-convex focal lens, and (iii) the optics tubbing holding all components of the optical system together. We played with two different approaches. We 3D printed the components for microscope but also bough with us from Chile the Edmund Optics optical tubes recommended at the open lab tools website .

3.1.-Objective

To magnify the sample, we used cheap microscope objectives widely available in the Chinese online market. We tested two types of objectives (**4x** objective and **10x** objective).We bought them on 淘宝网. They were shipped from Beijing and we received them at OpenFiesta 48 hours later.

This are cheap and good enough objectives. Other magnifications were widely available for every budget and quality. In general The objective are the easiest to find element of the optical system.

3.2.-Bi-convex focal lens

To focus the image (coming from the sample and captured by the objective) onto the Raspberry Pi camera sensor, we use a **bi-convex focal lens** with a of focal length of 50mm and with a diameter of 25mm. We used the focal lens made by Comar Optics **50VB25**[3]. This lens has AR coating blocking light except in the 450-900nm range. Blocking deep UV light is useful to protect the camera sensor when using blue light for illumination and excitation of fluorescent proteins. It is interesting however to also test the lens which are not coated. From Comar Optics catalog we can see they have one lens, product number **50VQ25** with this specs and which is a bit cheaper than **50VB25**. Ji Li (李季) has set himself to identify a local supplier in the local market for future builds of the microscope.

3.3.-Optical tube

Fancy optical tube. We have previously constructed a fancy optical tube as proposed by the Cambridge OpenLabTools site (*see* Fig. 6) and with parts (*see* Fig. 7) supplied by Edmund Optics[4]. In Shenzhen we played with it. This parts are a bit expensive however so an important cheaper version is to be consider. One were the tube components are 3D printed. At least two parts have to be printed. This are the Raspberry Pi Camara module mount and the

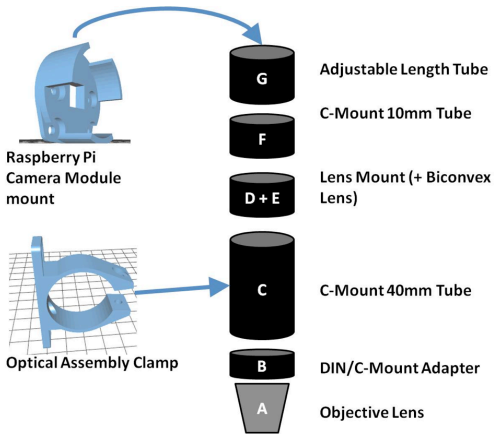


Figure 6. Optical tube components. *Edmund Optics*: (B) part #03-627, (C) part #54-631, (D) part #56-353, (F) part #54-629, (G) part #58-756; *Comar Optics*: (E) 50VB25 bicinvex lens; (A) an objective. (*Image from OpenLabsTool's original guide*).

[3] http://www.comaroptics.com/components/lenses/simple-convex-lenses/quality-biconvex-lenses

[4] https://www.edmundoptics.com/

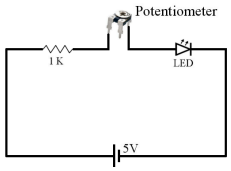
Optical Assembly Clamp. The Optical tube consists of: **(B)** a DIN/C-Mount mount to attach the DIN objectives to the rest of the tube which is C-Mount (part #03-627) and sales for $35USD, **(C)** C-Mount 40mm length tube (part #54-631) and sales for $35USD, **(D)** Lens mount for 25mm diameter (part #56-353) which sales for $73 USD, **(E)** is the 50VB25 biconvex lens which sales for $50 USD, **(F)** C-Mount 10mm length tube (part #54-629) which sales for $22 USD, and **(G)** C-Mount adjustable length, fine focus tube for the range 20mm to 30mm (part #58-756) which sales for $80 USD. Each of these parts can be ordered from Edmund Optics except part (E) which can be ordered from Comar Optics. As we can see from the prices at the date of writing, the optical tube cost around $300 USD. A heave price tag. However this prices are from Edmund Optics and Comar. Optics Could we find some local provider in Shenzhen? A very cheap alternative however, is to only buy the focal lens from Comar Optics and 3D print our own tube.

Figure 7. Optical parts,Edmund Optics. Parts needed to construct the Optical tube. Part numbers, photos from the Catalog for each component (G,F,D,C,B) of the tube shown in the previous figure.

3D printed tube At OpenFIESTA we 3D printed all the pieces provided OpenLabTools online repository and build a 3D printed optical tube. Tobey (黄园芳) took the lead and send all *.STL files to print at WeNext. From these 3D printed parts we constructed a 3D printed version of the optical tube connecting objective, focal lens, and camera sensor. All these files are available here and in the original repository from OpenLabTools. The difficult part however is that the documentation on *how to put it together* is not easy to find. We built it following our intuition and some online photos that Leslie Garcia and Paloma Lopez from Interspecifics took during a collaboration[5] with us at the Media Lab Prado (Madrid). The files needed to print the parts of an optical tube are described in the OpenFIESTA microscope **installation manual** and included on http://shenzhen.keymer.cl .

4.-Illumination

To illuminate the sample, we used a super simple (*see* Fig. 8) approach. We connected 5mm **white LED** attached directly under the sample to the *front* and *back* M6 rods making the stage frame with a 3D printer adapter designed by our friend and collaborator Fernán Federici while visiting Madrid with us for our co-collaboration with Interspecifics. Using this adaptor, the single LED is localized exactly under the hole we made in the stage cardboard. It is a very simple LED holder consisting of two files. The strength of the illumination was controlled by the microscope user using a potentiometer, the **+5V** and **GND** pins of the *Arduino* and a 1k Ohm resistor. We attach the LED to the adaptor by using female jumper wires.



Figured 8. LED circuit

[5] http://interspecifics.cc/comunicacionesespeculativas

5.-Moving the stage *up* and *down*: motor control via push buttons

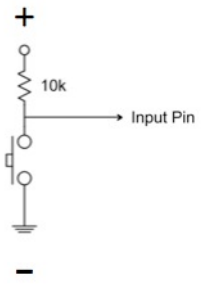


Figure 9. Push button circuit

In order to *focus* our imaging system (optically aligned objective, biconvex lens, & Pi Cam sensor; Section 3) on the sample of study (a cover slide on the microscope stage), we need to move the microscope stage (*Z-axis*) in the *up* and *down* directions. To accomplish this, the *linear translation axis*, embedded in the microscope Frame (Section 2) is powered by a stepper motor. We used a **Nema 17 stepper motor**[6] of 200 steps per each 360 degree full turn (1.8 degrees per step). We controlled the stepper motor using an ***EasyDriver***[7] stepper motor driver we bought in the Shenzhen electronic market (around 华强北 metro station) and which Alfredo (L'Homme) put to work in *no-time* after connecting pin headers to it to and hook it down to a breadboard we also got in 华强北. On the left side of this breadboard we connected the *EasyDrive*r to **+12V** and **GND** lines running on the top left of the breadboard to power the stepper motor. The four cables controlling the phases of the stepper motor were connected with pin jumpers to the corresponding four pins of the *EasyDriver*. The step and direction pins of the *EasyDriver* are connected to **pin9** and **pin8** respectively. On the right side of the breadboard we connected two *push buttons* via pull up (10 kOhm) resistors to the **+3.3V** power line on the *Arduino* board. When the push button is push, it grounds the circuit temporally by connecting the input pin on the Arduino to is own ground **GND** line (*see* Fig. 9). We use two of such buttons and wire them into **pin2** and **pin3** of the *Arduino* board.

5.1-The *Arduino* code

The code (*see* Box 1) was evolved to our needs by modifying previously existing open source code from the *Arduino* tutorials on push buttons[8] and stepper

```
#define DISTANCE 1

int StepCounter = 0;
int Stepping = false;

void setup() {
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
  digitalWrite(8, LOW);
  digitalWrite(9, LOW);

  pinMode(2, INPUT);
  pinMode(3, INPUT);
}

void loop() {
  if (digitalRead(3) == LOW && Stepping == false)
  {
    digitalWrite(8, LOW);
    Stepping = true;
  }

  if (digitalRead(2) == LOW && Stepping == false)
  {
   digitalWrite(8, HIGH);
    Stepping = true;
  }

  if (Stepping == true)
  {
 digitalWrite(9, HIGH);
    delay(1);
    digitalWrite(9, LOW);
    delay(1);

    StepCounter = StepCounter + 1;

    if (StepCounter == DISTANCE)
    {
      StepCounter = 0;
      Stepping = false;
    }
  }
}
```

Box 1. Micro-controller code to control de stepper motor (controlling linear motion) with *EasyDriver* via pin8 and pin9 on the *Arduino*. The machine listens to two push buttons pulled-up to HIGH and which ground the pin2 and pin3 to LOW when the user clicks on them to move the stage up or down.

[6] http://reprap.org/wiki/NEMA_17_Stepper_motor

[7] http://www.schmalzhaus.com/EasyDriver/

[8] https://www.arduino.cc/en/tutorial/pushbutton

motor control [9] using *EasyDriver*.

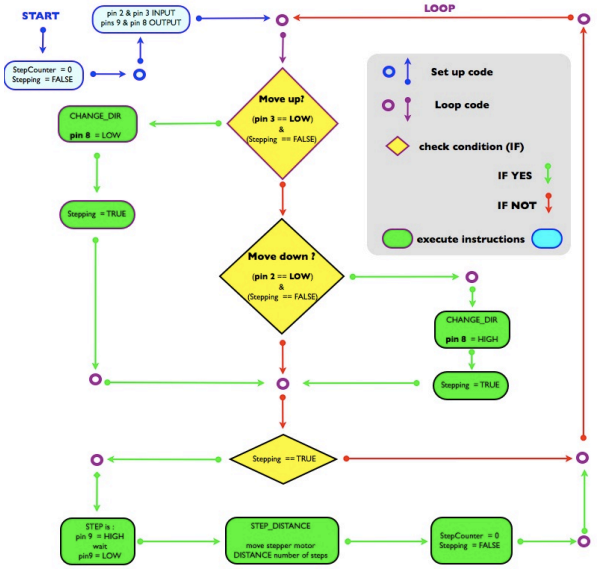
As global variables, we use a macro called DISTANCE that defines the number of steps to move when a button push (LOW signal on **pin2** or **pin3**) is detected (we use one single step). We also define to global flag variables (StepCounter and Stepping) to count the steps actuated when in action and to know performing or not the action of stepping. We start with no stepping and counter set to zero. Like all *Arduino* code it contains two sections.

In the **setup()** section section we define **pin8** and **pin9** as OUTPUT pins. Then we set these pins to LOW. We use these pins to send out motor actions (instructions) to the Easy driver controlling the stepper motor. We use two pins for two possible instructions (*motor driver actions*) STEP_DISTANCE, and CHANGE_DIR. Then we set **pin2** and **pin3** to act as INPUT pins. We use these pins to interact with the user via push down buttons.

In the **loop()** section, If all is ok, and the user is not pressing buttons, **pin8** and **pin9** are LOW and **pin2** and **pin3** are HIGH (due to *pull-up* resistors). While running, **loop()** checks for pin events via function digitalRead()**.** As global flag Stepping is se to FALSE and the push buttons are not being press the **pin2** and **pin3** are *pulled up* and both are HIGH. There fore the two if conditions on pins 2 and 3 lead to no further code being executed since no user is interacting with the machine.

5.2.-User interaction: moving the stage *up* and *down*.

To focus the optics on the sample, if the microscope user press a *push button*, the button being push will *temporally ground the pin* (*see* Fig. 9) attached to the button and the **loop()** code will detect it (*see* Box 2) while running one of the two first if functions of the code. If the ***move up*** button is pressed will be LOW signal on **pin3** and the code will set variable Stepping to a TRUE value as well as set **pin8** (direction change) to LOW using function digitalWrite() to tell *Easy driver* to not change the direction of the the direction of movement of the stepper motor (clockwise, CW). If a LOW is detected in **pin2** calling a ***move down*** signal handler, we set **pin8** to HIGH and reverse direction of movement (counterclockwise, CCW) telling *Easy driver* to change the direction of stepping (*motor driver action* CHANGE_DIR). The general consequence of either button triggering a push event is that the global variable Stepping is set to state TRUE.



Box 2. User interaction algorithm. Here we sketch the logical flux of the algorithm implemented by the code of Box 1.

The third if statement on the **loop()** function is now executed. This piece of code actually perform the step (*motor driver action* STEP_DISTANCE) on the direction set by the other part of

the code. The code tells the *EasyDriver* controller to STEP_DISTANCE which corresponds to a HIGH on **pin9** for 1 millisecond and then grounds it back to LOW to perform the step. After this, the Step code resets the SteppingCounter and sets Stepping to FALSE before exiting.

6.-Imaging system

Image/video capture as well as time lapse microscopy were all performed using a **Raspberry Pi** [10] computer and a **Raspberry Pi Camera**[11,12] whose lens were removed (*see* Fig. 10) in order to expose receptor array to the image formed by the biconvex lens (see Section 2) at a 50mm distance from the lens (separation the optical tube has between the lens and the Pi Camara holder).

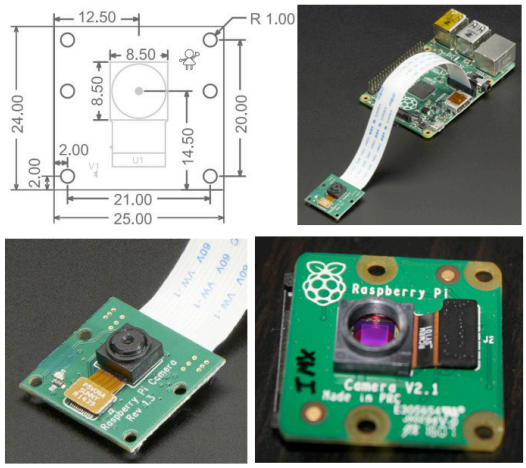


Figure 10. Raspberry Pi camera de-caping. Removing the lens of the *Raspberry Pi Camera* to expose the sensor. see text.

6.1.-Linux basics

Setting up the Raspberry Pi computer and the Pi Camera. We used NOOBS[13] as an easy way to get Linux running in no time on the Raspberry Pi computer we got in 华强北. We used a previously formatted SD card we had around. NOOBS is a boot up program which installs *Raspian*[14] as the operating system on the Raspberry Pi.

The Linux Terminal. The terminal application on the *Raspian* flavor of the Linux operating system are accessed via a Linux shell. If you are new to Linux (learn more), you can think of it as your agent program which allows you to type in commands and read out messages from the computer machine controlling the hardware you know as a Raspberry Pi.

Connecting to the microscope via Ethernet.We can access the microscopy from any other computer taking advantage of the fact that the Raspberry Pi is a network enable machine. Thus, if we connect an Ethernet cable from the Raspberry Pi Ethernet port to another computer we can run *ssh* to connect into the Raspberry Pi and run a remote terminal to control the microscope over the internet.

6.2.-Image capture and time-lapse microscopy

Raspberry Pi camera After having our Raspberry Pi Linux station up and running, make sure to *configure the camera module* by running *raspiconfig*! Notice we removed the lens of the Raspberry

[10] https://www.raspberrypi.org/

[11] https://www.raspberrypi.org/products/camera-module-v2/

[12] https://www.raspberrypi.org/documentation/raspbian/applications/camera.md

[13] https://www.raspberrypi.org/downloads/noobs/

[14] http://www.raspbian.org/

Pi camera. (know as de-cap the Raspberry Pi Camera[15]) to have a naked Raspberry Pi camera receptor exposed to the light coming out of our bi-convex lens focusing the image of the objective into the Raspberry Pi camera receptor.

Taking pictures with *raspistill* To take pictures with the raspberry Pi camera we type the *raspistill* [16] command on the Linux shell from the Raspberry Pi. To take a picture from the camera simply type:

```
>> raspistill -o camera_capture_picture.jpg
```

into the Raspberry Pi Linux terminal. To create am image file in jpg format and named "camera_capture_picture.jpg". To see more options on this command please study from the Raspberry Pi documentation .

Time lapse microscopy: capturing images. To make a time lapse video from a sequence of pictures, we first need *a collection of pictures* taken at successive intervals of time.Program *raspistill* has a built in option to perform such task. Issuing the following commands at the shell

```
>> cd ~
>> mkdir My_Timelapse_Pictures
>> cd My_Timelapse_Pictures
>> raspistill -o a%04d.jpg -t 3600000 -tl 30000
```

we will move to the home directory of the user pi (assuming we logged in as the default user. User pi) and then we make a directory called My_Timlapse_Pictures. Then we move inside the directory and run the *raspistill* command every 30000 millisecs (30 seconds) for 3600000 millisecs (1 hour) creating a picture filein jpg format with a name determined by the timestamp (`a%04d`) in the file name argument.

Time lapse microscopy: generating a time lapse movie. Once a directory with the time-lapse pictures is generated, this can be concatenated into a single movie. For example, for the directory called My_Timlapse_Pictures previously generated, we can issue the following command

```
>> cd ~/My_Timelapse_Pictures
>> ls *.jpg > stills.txt
>> mencoder -nosound -ovc lavc -lavcopts
   vcodec=mpeg4:aspect=16/9:vbitrate=8000000
   -vf scale=1920:1080 -o timelapse.avi
```

using the MEncoder library[17] to generate timelapse.avi file with a video recording of the timelapse.

Video streaming from the imagining system with *raspivid.* An alternative method, is to capture a video stream directly from the Raspberry Pi camera using the command *raspivid*[18].

---

[15] https://www.raspberrypi.org/forums/viewtopic.php?f=43&t=148929

[16] https://www.raspberrypi.org/documentation/usage/camera/raspicam/raspistill.md

[17] https://help.ubuntu.com/community/MEncoder

[18] https://www.raspberrypi.org/documentation/usage/camera/raspicam/raspivid.md

```
>>raspivid –t 0
```

will, stream video until the process is killed. Using *raspivid* command we can record video into a file using different video file formats as well as stream video from the camera. A useful tip is to notice that raspivid has numerous options controlling exposure and other parameters. As we want to control the light intensity with a petenciometer we do not want the Raspberry Pi camera to adjust exposure on the run. For this we can use the following command

```
>>raspivid –t 0 –ex off –v –awb sun
```

which makes the camera video stream be captured with no exposure and no awb. If we want to control the screen area and location of the video stream, we can use

```
>>raspivid –t 0 –ex off –v –awb sun –p '10,10,400,400'
```

We can see the great power of the commands *raspivid* and *raspistill* to control the video captured system at the core of our imaging system.

7.-Discussion & Outlook for *version* 2.0

Re-design the 3D printed T-bracket and Shaft clamps. In the interest of time we re-cycle 3D printer files which were originally designed for the Cambridge microscope and thus made to fit onto an OpenBeams based frame. This parts, work on relatively well on a V-slot after removal of material on the holes made to fit the thicker screws needed to attach nuts and screws to the V-slot. Will be of use to design new files made to fit theV-slot configuration. These parts are: (i) The T OpenBeam brackets used to secure the small beams to the upper and lower 2D frames making the top and bottom parts of the frame, and (ii) the shaft clamp holding the two smooth rods acting as linear guides for the stage up and down movement. The optical system parts must be also be search in the local ecosystem to create a prime version where the optics are not printed 3D as well.

V-slot stepper motor bracket. Another issue to consider is the stepper motor bracket to hold the motor to the beam. We used an OpenBeam bracket we hacked by enlarging the screw holes since we had one extra at hand and wanted to save time to survive in the Shenzhen speed ecosystem. The V-slot bracket, is slightly different and therefore a modification at the base and top bases is needed. the easies is to move the small beam to the back end of the bases or to replace a beam for a double V-slot beam.

Arduino* and *Raspberry Pi* communication**. Another improvement is to add communication between the imaging system running on the Pi and the motor control system running on the Arduino controller. An easy way to achieve this is to use the ***gpio[19] utility of the ***Wiring Pi***[20] library to talk to Arduino pins via a logic level convertor from the generic input output pins on the *Raspberry Pi* board. Another way to communicate is via Serial Communication. This is needed in case more actuation is needed. For example to implement auto focus algorithms or communications with a

[19] http://wiringpi.com/the-gpio-utility/

[20] http://wiringpi.com/

more capable stage.

Evolving the microscope frame. After moving from *OpenBeams* to 20x20 *V-slots* we learned about numerous efforts in open source hardware dealing with *XYZ* movement. This mechanical machines (*see* Fig. 12) can to be exploited better to build *Z-axis* translation systems for focusing imaging capture camera devices and light sources over a biological samples. An obvious improvement to the current design. is to implement mobility in the *X* and *Y* axes of the stage. This will add controlled movement to the Stage allowing for *XY* scanning of sample. This is improvement is of fundamental importance. The strategy at hand is to learn from the 3D printer community and use CoreyXY and H-Bot XY-cartesian[21] systems as well *Z-XY robotic cubes*[22] to align imaging systems with focusing system (*Z-axis*) and robotic stages (*XY-axes*) capable of automatically sample the sample in space and take spatially shifted images in order to render large areas at high resolution. This way we could capture take spatial time series data.

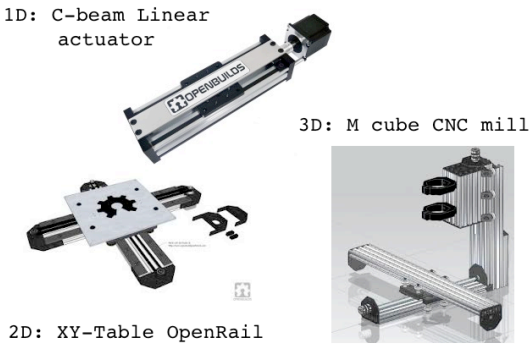


Figure 12. Modular Mechanics. One, two, and three dimensional movement systems are evolving in hardware space. The idea is to reach out to these communities and learn from them when building more capable open microscopes

Illumination system. The illumination system considered is very basic. Another improvement to the microscope would be to work out a better more versatile illumination system. So, far is only one type of light (white) and only one method (transilluminating from under the sample). Alternative sources of light, can be brought onto the sample via cheap fiber optics. It will be interesting to see this possibility. Also, will be interesting to test how illumination from the top will work. For example light coming from a LED ring around the optical tube.

Optical system. More study and practice on assembling the optical tube made of 3D parts. Documentation on this step is badly needed. At the same time we think that the optics deserves to be of better quality. Therefore we think an interesting venue for further improvement will be to find alternative providers of the C-mount tubes. The $300USD cost estimated at the time of writing was made considering Edmund Optics and Comar Optics as providers. We feel the 深圳创新生态系统 is capable of providing the same parts for a cheaper prize. An immediate goal is to find such providers and to implement a "Fancy-*but*-山寨" version of the optical tube.

Building the microscope has once again shown us the pleasure we find in *learning by doing*. 我们 sincerely hope to see how Shenzhen, its 创新生态系统, will make further iterations of the 机 and take DIY 生物 Open 显微镜 hardware to the next evolutionary state.

[21] http://openbuilds.org/builds/openxy.2911/

[22] http://openbuilds.org/?category=h-bot-and-core-xy&id=274

7.-Institutional acknowledgments

We thank the following organizations that helped financed this transpacific adventure:

OpenFIESTA for *hosting us* during an amazing two weeks of learning and growing together at Tsinghua University (清华大学) Shenzhen (深圳) Graduate School (研究生院).

FONDECYT 1150430 from Conicyt Chile supporting the Keymer laboratory's research and making our visit to Shenzhen possible.

Without these two efforts supporting us, the hackathon would of have been impossible, 谢谢大家!