# *IEROM – Operational Manual*

## *IEROM Startup*

We assume that there are four computing systems in this setting.

1. Main Control and Database Server – **cerebellum** – Local IP address: 110
2. Image Server – **imageServer** – Local IP address: 120
3. Image Processing Server – **brainslave** – Local IP address: 130
4. Microscope Control PC – **blackscope**

## *Power On for Stage and Camera*

In the power strip, all components of the system are clearly label. Fine STGE (stage) and CAM (camera). Then turn them ON.

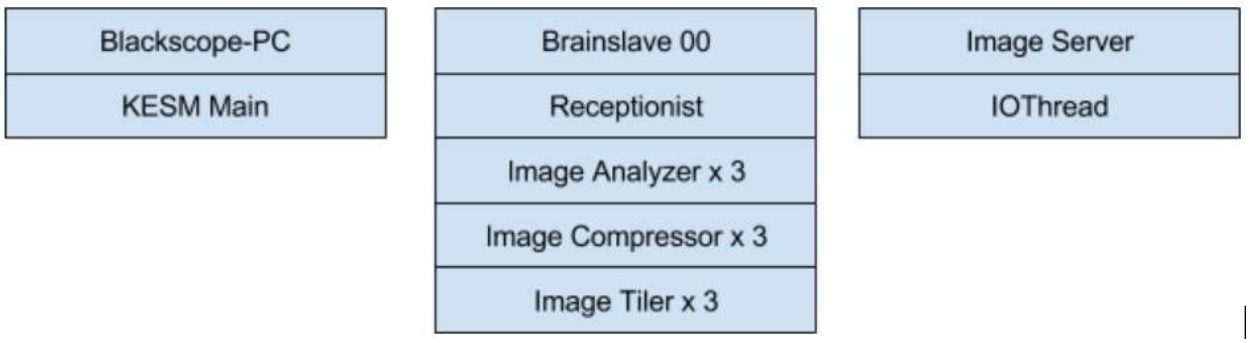
## *Software Startup*

- At cerebellum
    - Login as bi2s
    - Start a Terminal
    - Start a conda environment, named as **scope** for the system
        - $ source activate **scope**
    - Start the admin program
        - $ runscope -s admin
    - Open another Terminal
    - Start the control program with the name **black**.
        - $ source activate scope
        - $ runscope -s black
- At imageServer (IOThread will be running)
    - Login as bi2s
    - Start a Terminal
    - Start a conda environment and run the control program.

        - $ source activate scope
        - $ runscope
- At brainslave (ImageAnalyzer, ImageComposer, ImageTiler will be running)
    - Same as imageServer.
- At blackscope
    - Start Windows PowerShell
    - Go to C:\Users\BlackScope\
    - Use commands as follows.
        - activate scope
        - runscope

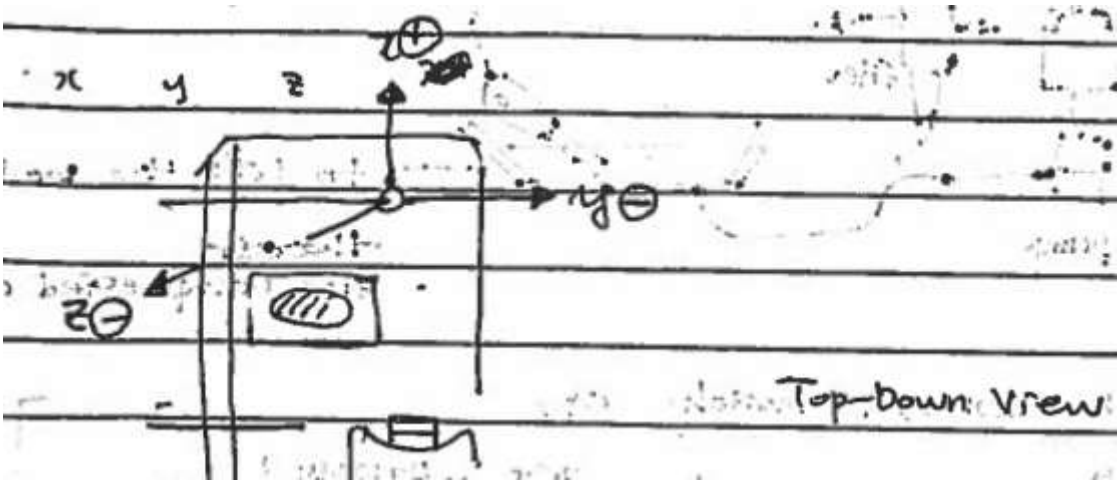
## *User Interface Startup*

- Open Chrome Browser.
- Use the URL below for admin.
    - http://admin.bi2s.ddns.net
- Use the URL below for the black microscope control
    - http://black.bi2s.ddns.net

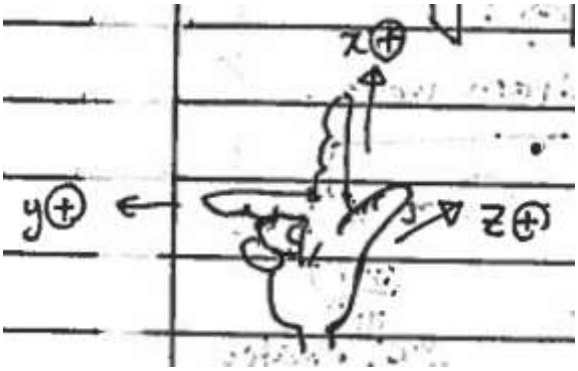


# *Initial Settings for Image Sectioning*

## *Homing*



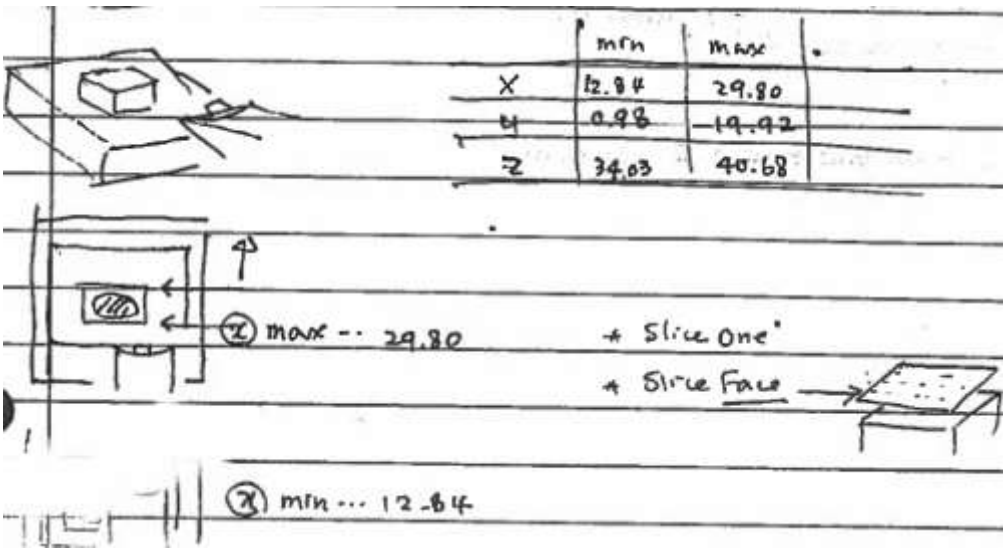

The axes of the system are as follows.

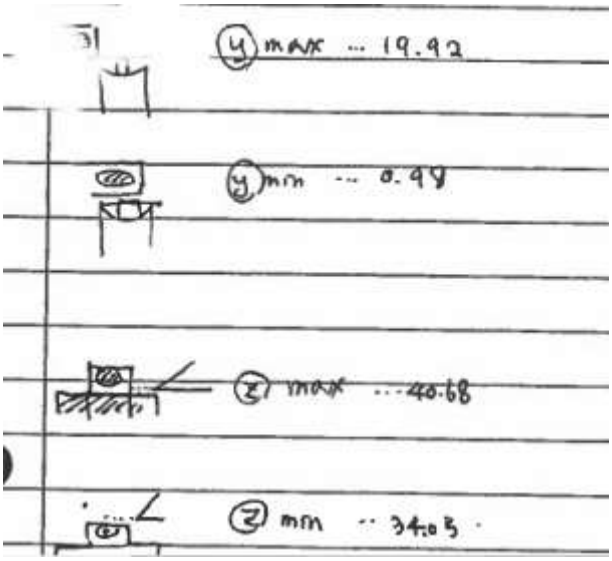



- Caution
  - Water pump must be off before homing!!
  - No knife attached!!
- Sign-in from .110 with localhost:3000
- Place the stage x, y, and z in somewhere in the middle.
- Homing order:
  - Enabling
    - Down z all the way.
      - Enable z → will be Fault.
        - Then click ‘Clear’ button.

        - Then z will be the lowest possible.
    - Enable x and y
- Homing
    - Home y first.
    - Then x.
        - Caution: Make sure that the knife will not hit the tissue block.

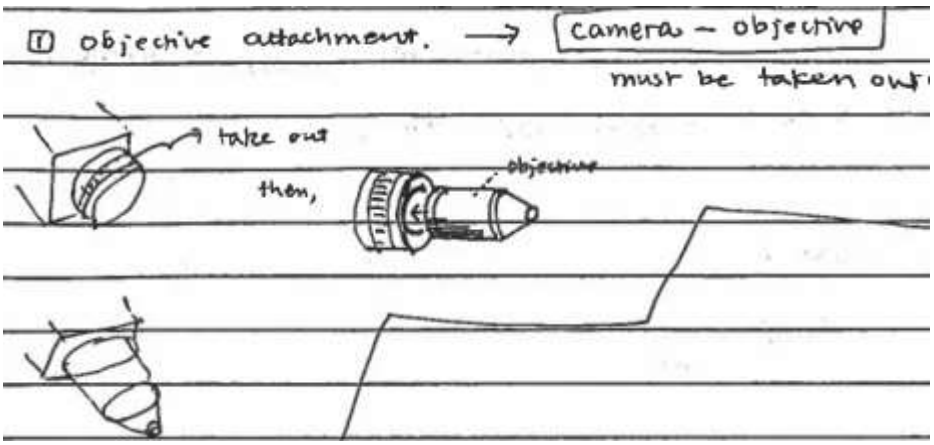
## *Tissue Block Size Measurement*

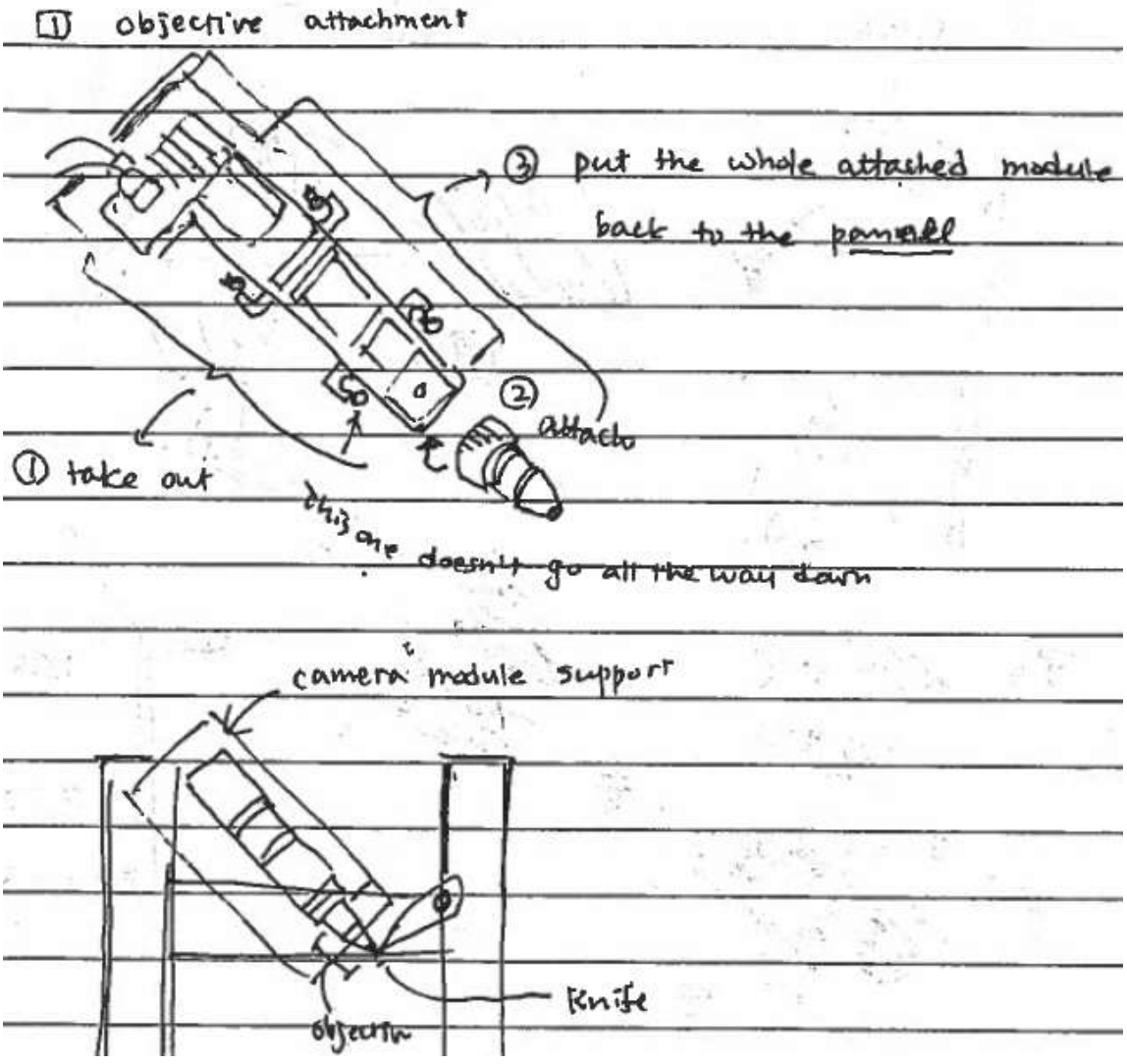


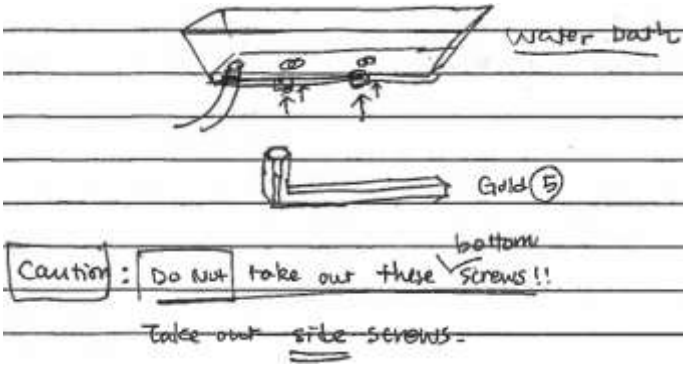
## *How to Attach an Objective*

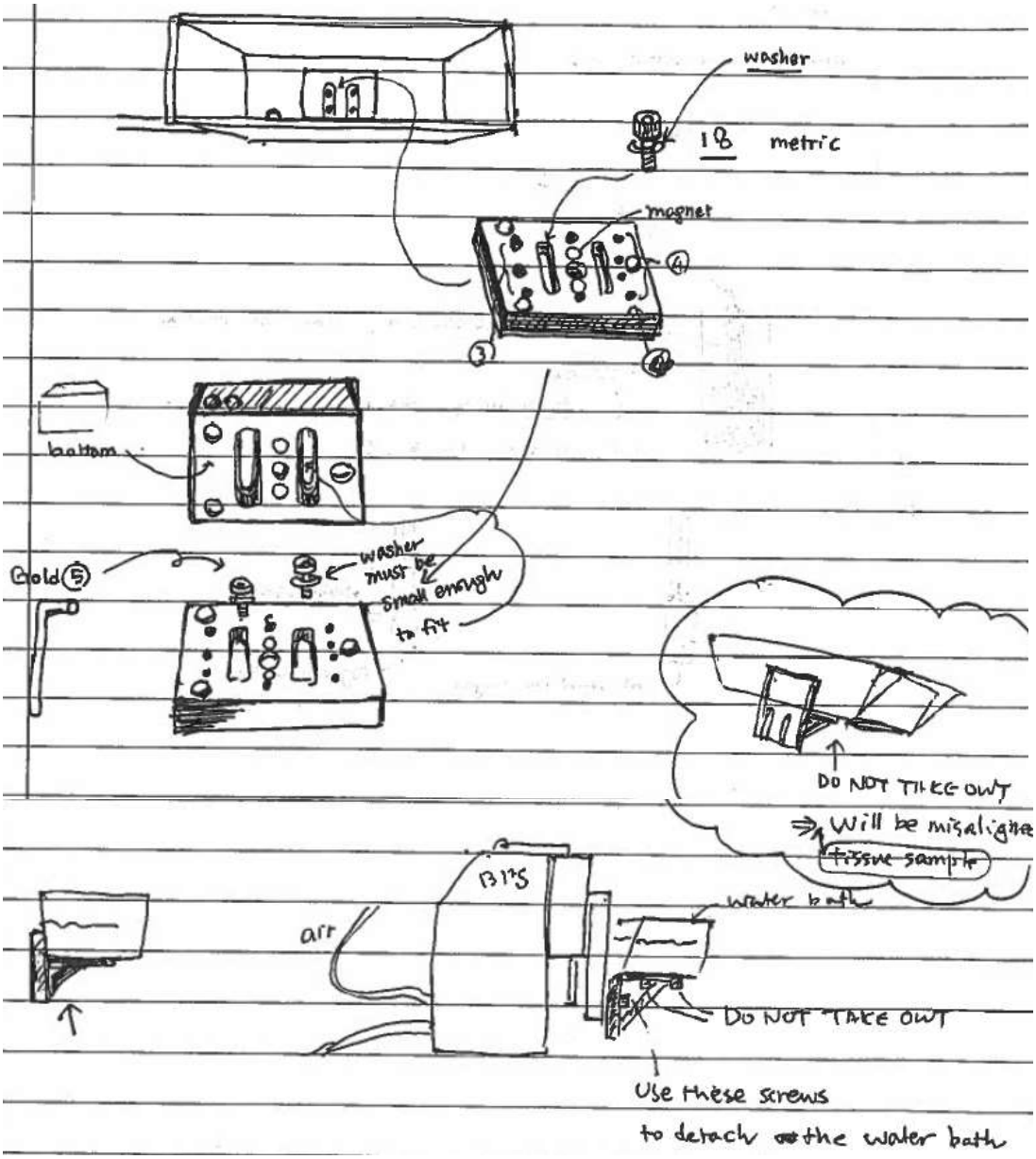
To attach the objective, the circular shape in the optics train must be taken off.



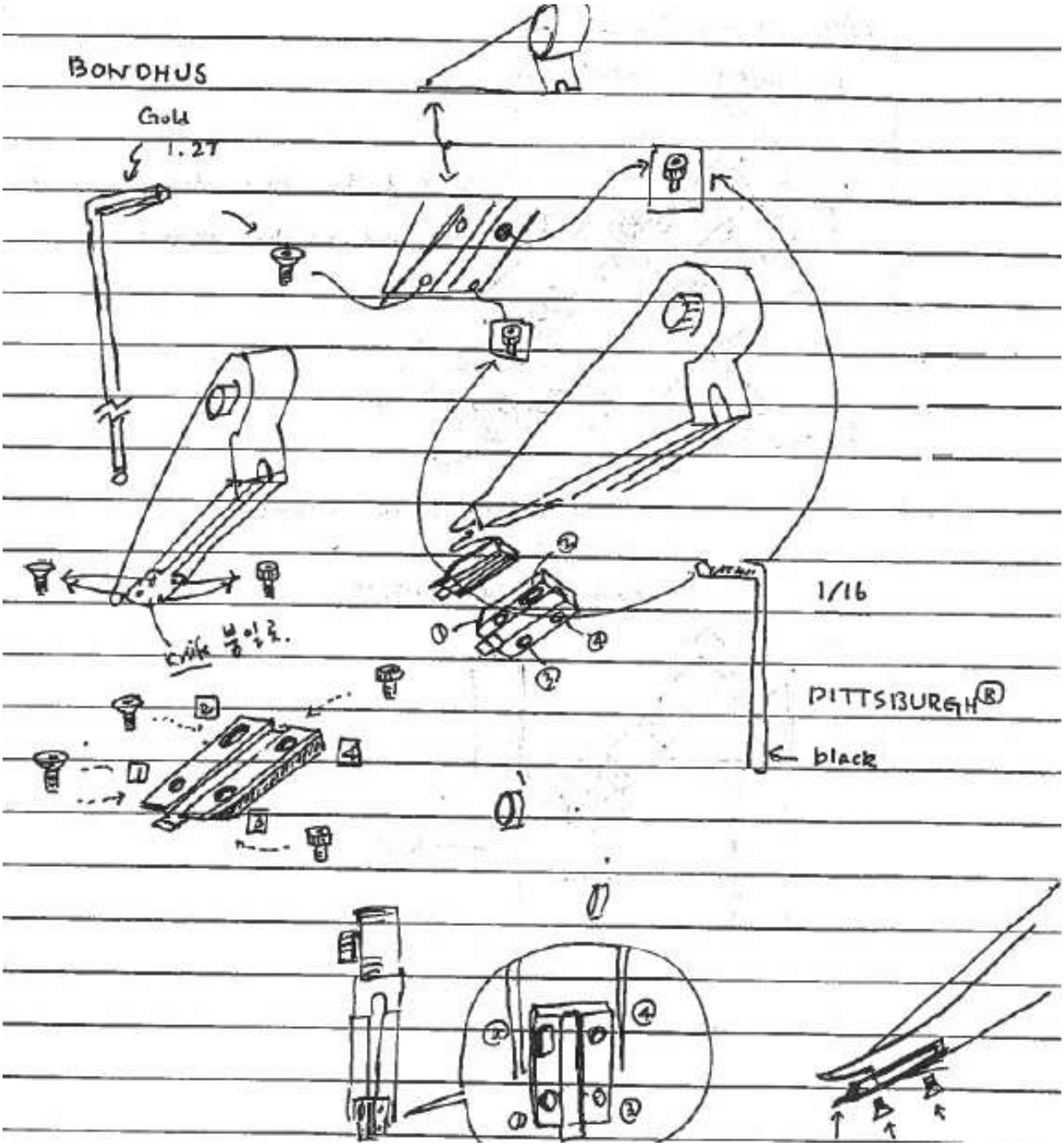



## *How to Disassemble Water Bath*

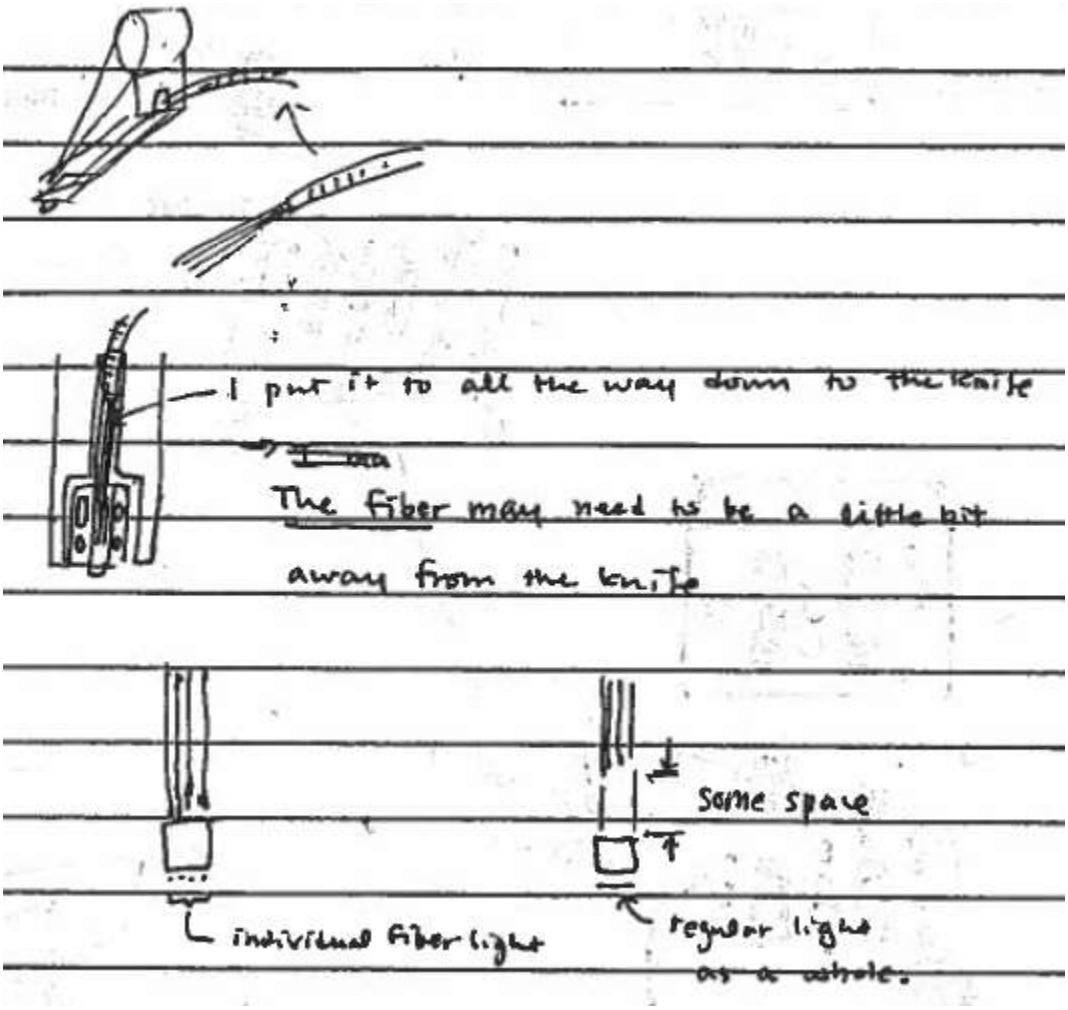


***How to Assemble the Diamond Knife***



## *How to Attach the Optic Fiber and Seal It*




## *Trouble Shootings*

When the stage goes mad, the enable/disable flag in the GUI is not responding. The flag must be manually changed in the database.

### *Robomongo:*

Manually change stage data.

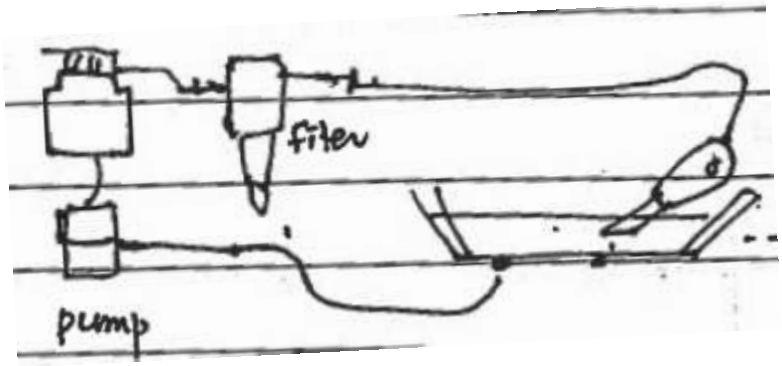
kesm-metor

- system
- Meteor - collections - properties-black
    - Right click and select ‘View Results in Table Mode.’
    - Find ‘Property’ column.

- Right click and select ‘ Edit Document’
    - “requestedValue” ← change this same as “value”

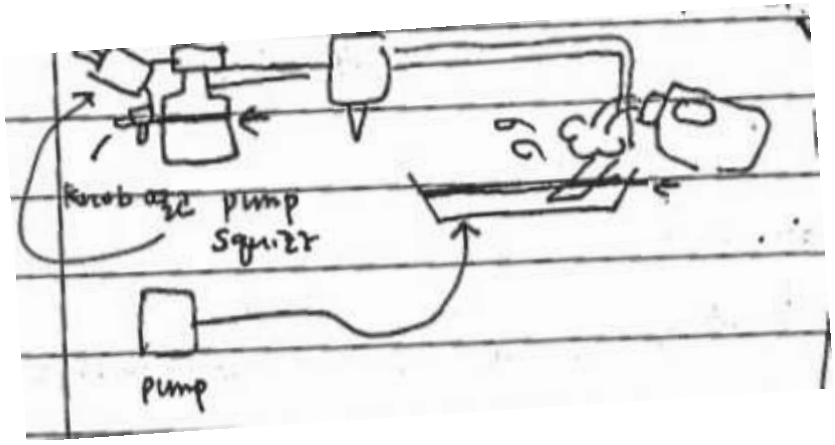
## *Water Pump*

The water bath must be lifted a little bit from the bottom. In order for the stage to hold up the water bath, the compressed air must be provided. See the Air Compressor section.



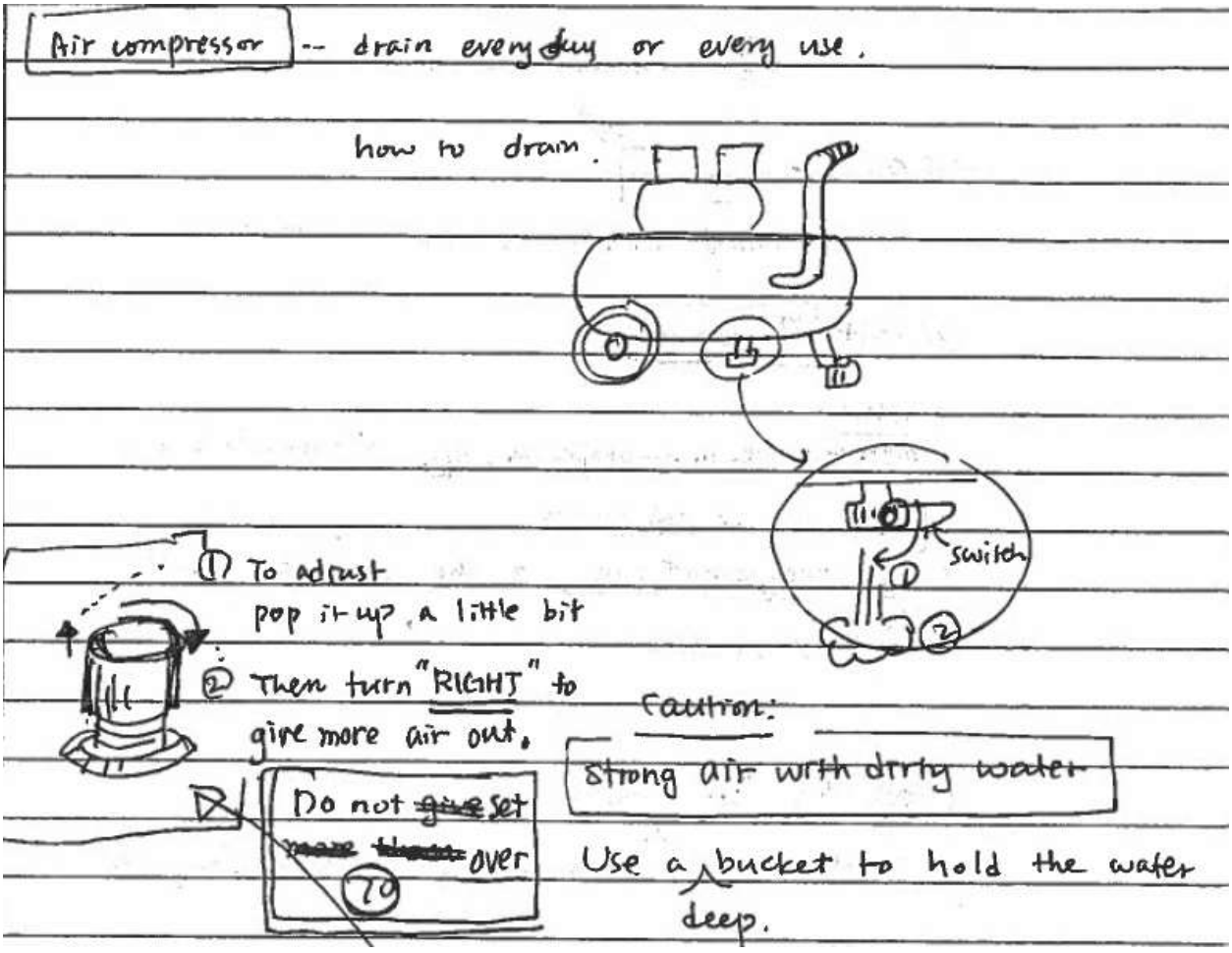

## *How to Adjust Water Level*

- Pour water into the water bath to the marked level.
- Lift the water bath to emerge the tip of the knife where the inlet is located.
- Turn on the water pump to circulate the water through the system.
- As the water level goes down, pour water a little by little.
    - Caution! Do not rush. Be careful not to overflow the water.
- Open the knob of the squeezer that is located in the top left corner of the drawing. Then repeat pressing and releasing to suck water to the bottle. There is a marked level on the bottle. You can safely get the water up to the level.

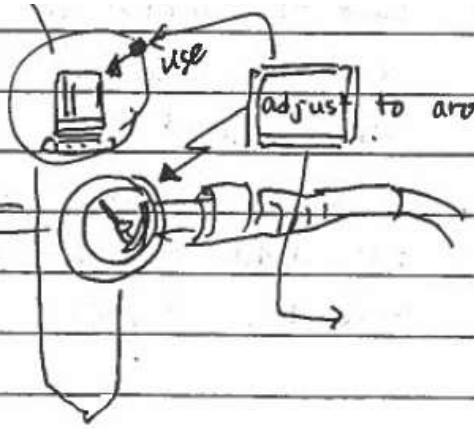



## *Air Compressor*

If the compressor does not give an enough PSI, then the water inside must be drained.



Air adjustment




must below the red range.

From this section, an operator does not need to do or use them for daily operations of the IEROM.

# *Network Settings*

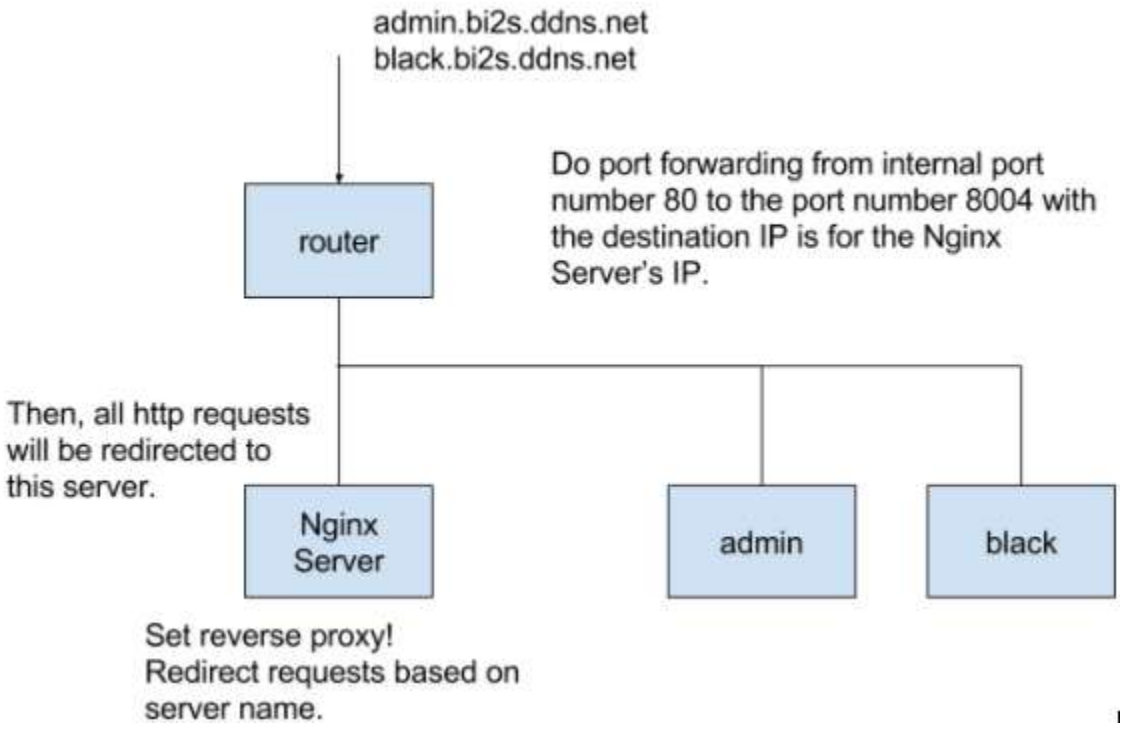
Networks must be properly set before any of the IEROM's software.

## *Subdomain name resolution*

### *.3scan-kesm-config.json file settings*

Remove the "baseURI" section in the meteor. There are "admin" and "black" section in the "meteor" section. Make sure that the "baseURI" section is removed from each of them.

## *Network configuration*




### *Use Nginx*

Reverse proxy. Use proxy_pass in nginx.conf file. (/etc/ngnix/conf.d/ngnix.conf)

Create a file named “ierom.conf”

```
# systemManager Meteor Server
server {
   listen       8004;
   server_name  admin.bi2s.ddns.net;


   location / {
       proxy_pass http://192.168.1.110:3000;
   }
   proxy_http_version 1.1;
       proxy_set_header Upgrade $http_upgrade;
       proxy_set_header Connection "upgrade";


}


# kesmControl-black Meteor Server
server {
   listen       8004;
   server_name  black.bi2s.ddns.net;


   location / {
       proxy_pass http://192.168.1.110:3100;
   }
   proxy_http_version 1.1;
       proxy_set_header Upgrade $http_upgrade;
       proxy_set_header Connection "upgrade";


}


## must be completed with data and file servers. Here!!! 5/4/2017
```

### *Start/Stop nginx*

http://nginx.org/en/docs/beginners_guide.html

nginx -s *signal*

Where signal may be one of the following:

- stop — fast shutdown
- quit — graceful shutdown

- reload — reloading the configuration file
- reopen — reopening the log files

## *Fix the error, "Uncaught Error: Handler with name 'route' already exists."*

This is from iron:middleware-stack in meteor module.

### *Workaround without updating iron:middleware-stack*

http://stackoverflow.com/questions/36031706/middleware-stack-js31-uncaught-error-handler-with-name-route-already-exists

> dit: this issue was fixed in iron:middleware-stack 1.1.0 .
>
> I have the same problem. Weirdly, I have this problem on Chrome 51 but not on Chrome 46. I guess this has to do with updates in the javascript engine, and I'll post here if I figure out what exactly.
>
> In the meantime, the workaround I used was to explicitly add names to the routes. It doesn't matter what they are, they just have to be declared, otherwise iron-router think the name of the route is "route." So your code would become:
>
> ```
> Router.route('/admin/dashboard', { name: "Boaty_McBoatface",
> template:"adminDashboard" }); Router.route('/admin/create/table', { name:
> "Guacamole", template:"create_table" });
> ```

### *My fix!*

Based on the idea quoted above, I changed router.js files.

router.js files are located in each meteor web directory. In our case, there are at followings.

~/anaconda3/pkgs/3scan-scope-0.5.0-nppy_35/site-packages/KESMAcq-0.5.0-py2.7.egg/KESMAcq/web

kesmControl/client/router.js

systemManager/client/router.js

Due to the error in iron::middleware-stack, the definitions of Router.route must have an explicit name.

Thus, I added 'name: <any_name>.' Without the name property, all names are set as 'route' as default.

This makes any additional Router.route generate an error, "***Uncaught Error: Handler with name 'route' already exists.***"

---

*kesmControl/client/router.js*

```
var titlePrefix = 'KESM@KU-' + kesmName + ' ';


Router.route('/', {
 name: 'root',
 template: 'kesmControls',
 waitOn: function() {
   return [
     Meteor.subscribe('current-properties'), // this includes the navbar-
properties
     Meteor.subscribe('new-images-from-kesm', kesmName, 50)
   ];
 },
 onAfterAction: setPageTitle(titlePrefix + 'Control')
});


Router.route('/focus', {
 name: 'focus',
 template: 'focusViewer',
 waitOn: function() {
   return [
     Meteor.subscribe('kesm-control-navbar-properties'),
     Meteor.subscribe('focus-image', kesmName)
   ];
 },
 onAfterAction: setPageTitle(titlePrefix + 'Focus Viewer')
});


Router.route('/debug', {
 name: 'debug',
 template: 'debugPane',
 waitOn: function() {
   return [
     Meteor.subscribe('kesm-control-navbar-properties'),
     Meteor.subscribe('current-properties')
   ];
 },
 onAfterAction: setPageTitle(titlePrefix + 'properties')
});


Router.route('/stage', {
 name: 'stage',
 template: 'stageData',
```

```
 waitOn: function() {
   return [
     Meteor.subscribe('kesm-control-navbar-properties'),
     Meteor.subscribe('stage-data-for-this-kesm', kesmName)
   ];
 },
 onAfterAction: setPageTitle(titlePrefix + 'Stage Data Viewer')
});
```

---

*systemManager/client/router.js*

```
/* globals Router, setPageTitle */


Router.route('/', {
 name: 'root',
 template: 'dashboard',
 waitOn: function() {
   return [
     Meteor.subscribe('multi-kesm-properties'),
     Meteor.subscribe('shared-properties')
   ];
 },
 onBeforeAction: function() {
   Session.set('numLogs', 6);
   Session.set('logSeverity', 'INFO');
   Session.set('moduleSearch', '');
   this.next();
 },
 onAfterAction: setPageTitle('KESM Dashboard - Kettering')
});


Router.route('/processes', {
 name: 'processes',
 template: 'processes',
 waitOn: function() {
   return [
     Meteor.subscribe('allowed-processes')
   ];
 },
 onAfterAction: setPageTitle('KESM Processes')
});
```

```
Router.route('/logs', {
 name: 'logs',
 template: 'logViewer',
 waitOn: function() {
   return [
     Meteor.subscribe('all-log-device-names'),
     Meteor.subscribe('shared-properties')
   ];
 },
 onBeforeAction: function() {
   // In case we're coming from the dashboard, make sure this is populated
correctly.
   if (Session.get('numLogs') < 100) {
     Session.set('numLogs', 100);
   }
   this.next();
 },
 onAfterAction: setPageTitle('KESM Log Viewer')
});


Router.route('/slices/sample/:sampleId/z/:z', {
 name: 'slice-viewer.image',
 template: 'sliceViewer',
 waitOn: function() {
   var options = {
     sampleId: this.params.sampleId,
     // We've gotta parse a float here because route params are
     // always strings.
     z: parseFloat(this.params.z)
   };
   return [
     Meteor.subscribe('tiled-images-in-face', options),
     Meteor.subscribe('image-in-next-face', options),
     Meteor.subscribe('image-in-previous-face', options),
     Meteor.subscribe('image-at-lowest-z', options),
     Meteor.subscribe('image-at-highest-z', options)
   ];
 },
 data: function() {
   var routeQuery = this.params.query;
   var lat = parseFloat(routeQuery.lat);
   if (_.isNaN(lat)) {
     lat = 0;
   }
   var lng = parseFloat(routeQuery.lng);
   if (_.isNaN(lng)) {
     lng = 0;
   }
```

```
    var zoom = parseInt(routeQuery.zoom, 10);
    if (_.isNaN(zoom)) {
      zoom = 0;
    }


    var sampleId = this.params.sampleId;
    var z = parseFloat(this.params.z);


    return {
      sampleId: sampleId,
      z: z,
      lat: lat,
      lng: lng,
      zoom: zoom
    };
  },
  onAfterAction: setPageTitle('KESM Slice Viewer')
});


Router.route('/slices/sample/:sampleId', {
  name: 'slice-viewer.sample',
  action: function() {
    var sampleId = this.params.sampleId;
    this.redirect('slice-viewer.highest', { sampleId: sampleId });
  }
});


Router.route('/slices/sample/:sampleId/no-images', {
  name: 'slice-viewer.no-images',
  template: 'sliceViewer',
  waitOn: function() {
    var sampleId = this.params.sampleId;
    return [
      Meteor.subscribe('latest-tiled-images-for-sample', sampleId, 1)
    ];
  },
  data: function() {
    return {
      sampleId: this.params.sampleId,
      imagesInFace: Images.find().count()
    };
  }
});
```

```
Router.route('/slices', {
 name: 'slices',
 waitOn: function() {
   return [
     Meteor.subscribe('latest-tiled-image')
   ];
 },
 action: function() {
   var lastImage = Images.findOne();
   this.redirect('slice-viewer.sample', { sampleId:
lastImage.properties.currentSampleID });
 }
});
```

## *Websock Error*

### *WebSocket connection failed: Error during WebSocket handshake: Unexpected response code: 400*

https://chrislea.com/2013/02/23/proxying-websockets-with-nginx/

In nginx settings, I added three lines in location / {} section.

```
server {
  ….
  location / {
     proxy_pass http://localhost:8080;
     proxy_http_version 1.1;
     proxy_set_header Upgrade $http_upgrade;
     proxy_set_header Connection "upgrade";
     proxy_set_header Host $host;
  }
}
```

proxy_http_version 1.1;
proxy_set_header Upgrade $http_upgrade;
proxy_set_header Connection "upgrade";
“The first line tells Nginx to use HTTP/1.1 when communicating to the Node backend, which is required for WebSockets. The next two tell Nginx to respond to the Upgrade request which is initiated over HTTP by the browser when it wants to use a WebSocket.

In production, you’d likely want to add additional location stanzas to Nginx to tell it where to serve static assets from, set expires headers, and so on. You’d also likely want to manage the Node process(es) with an init script or supervisor, so that the app would start automatically

when the server booted up. But, in an nutshell, this is pretty much it for using Nginx with your WebSocket enabled application! Questions and comments are always welcome of course." [from the link]

### *DevTools failed to parse SourceMap:*

### *http://black.bi2s.ddns.net/bootstrap.css.map*

In DevTools, press F1. Turned off two options. "Enable Javascript source maps" and "Enable CSS source maps."

http://stackoverflow.com/questions/36051891/esri-failed-to-parse-source-map