OzturkLab Report

Peker Milas

December 7, 2020

Contents

1 Self-recovering Wired Ethernet Connection

This is a software based workaround for our lab's computers to keep the in the wired network. To implement the solution, following software is required to be installed;

- **Chrome** web browser
- **Anaconda** or another Pyhton distribution.
- **Selenium** libraries for Anaconda to be able to interface with Selenium.
- **Selenium** Web Driver

1.1 Preparation and Software Installations

1.1.1 Chrome Installation

To install Chrome, go to following web page and download the suitable installer version for your operating system (OS). [1]

1.1.2 Anaconda Installation

To install Anaconda, go to web page and download Anaconda installer suitable for your OS. [2]. During the installation, at some point installer will give you an option to add Anaconda to system path ($PATH) as shown in Figure1.

[1] `https://www.google.com/chrome/`
[2] `https://www.anaconda.com/products/individual`

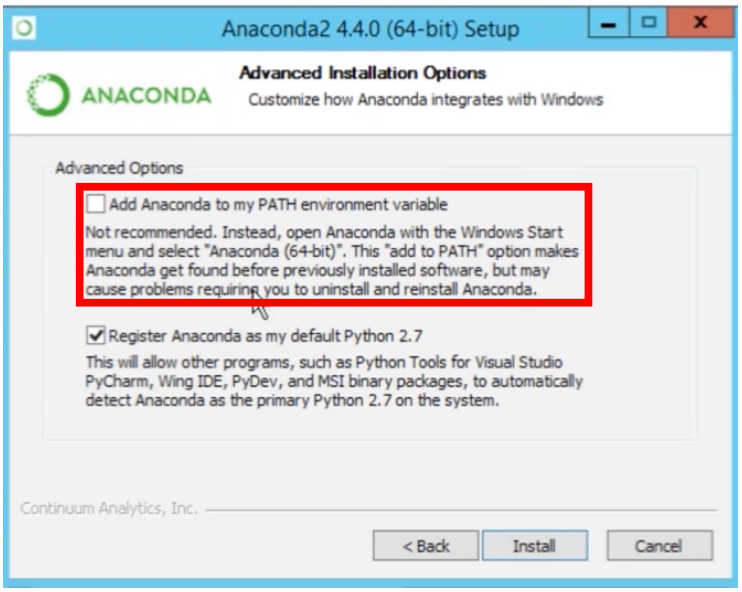

Figure 1: Installation step for adding Anaconda to system path.

Here, go ahead and click the box on the left to add Anaconda to system path. This will make Anaconda your system-wide Python library. Its issues are not going to be discussed here for simplicity.

1.1.3 Selenium Library Installation

To install Selenium libraries for allowing Anaconda to interface with Selenium web driver you can use Anaconda Prompt as shown in Figure2.

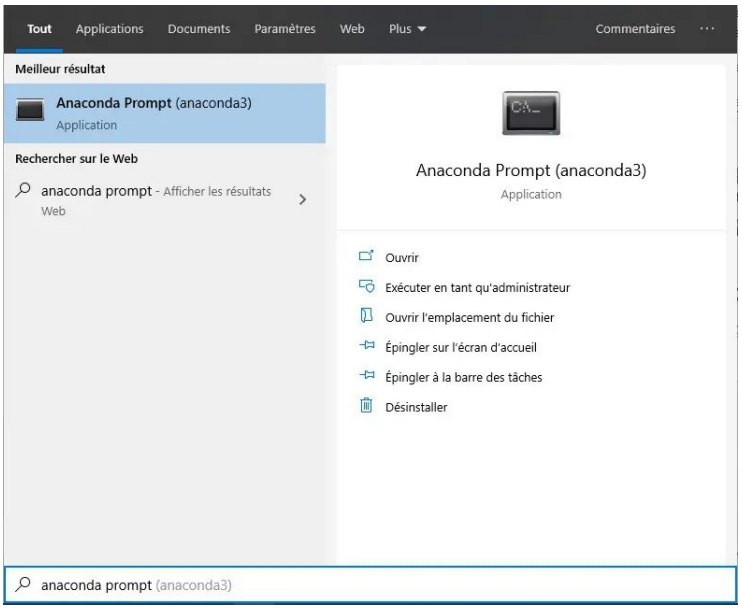

Figure 2: Opening up Anaconda Prompt.

In Anaconda Prompt, you can simply type **conda install -c conda-forge selenium** and click **Enter**, as shown in Figure3.

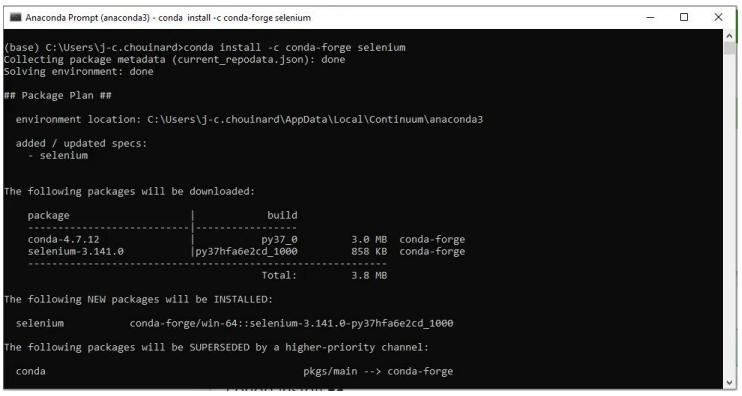

Figure 3: Selenium library installation for Anaconda.

Once Anaconda figures out where Selenium library is and possible additional required libraries, it will ask your confirmation as shown in Figure 4. Here you can type **y** and click **Enter**.

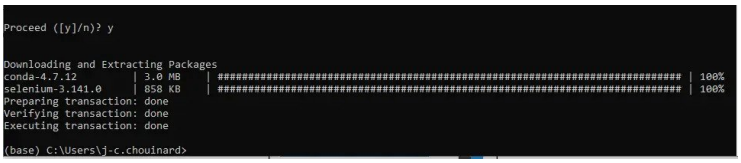

Figure 4: Selenium library installation confirmation screen.

1.1.4 Selenium WebDriver Installation

To install Selenium WebDriver for allowing Selenium-Anaconda library to control the web browser (**here we use Chrome!!!**), go to the following web page, `https://sites.google.com/a/chromium.org/chromedriver/` as shown in Figure 5.

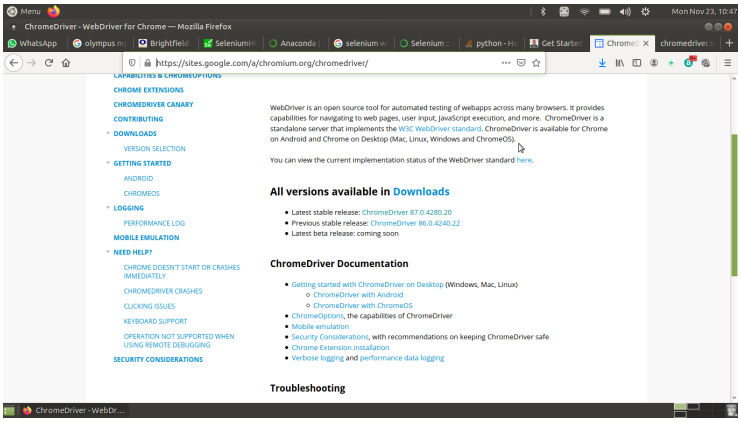

Figure 5: Selenium library installation confirmation screen.

Within other options, click on the **Latest Stable Release**. This will direct you another web page with multiple files you can download as shown in Figure 6. **Here, you have to download 32-bit driver!!!**

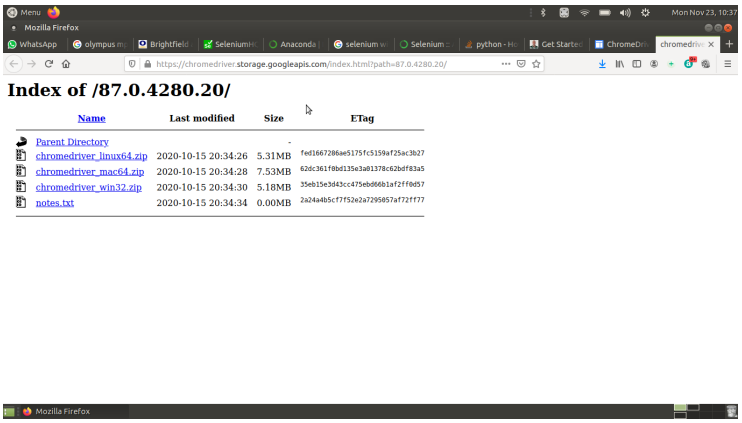

Figure 6: Selenium library installation confirmation screen.

Once download finishes, you can unzip the the webdriver file with any software you desire.

1.2 Implementation-Coding

To invoke Selenium I wrote Python Script. An overwiev of the code is as following;

1. **Lines 1-4:** Imports essential libraries.
2. **Lines 6-8:** Pings Google's DNS server to check whether the PC is connected to wired network.
3. **Lines 10-11:** If output of lines 6-8 are both indicating connection proceed to the end of the code.
4. **Lines 12-27:** If PC is not connected; **Line 13:** Run Chrome driver **Line 14:** Send Chrome to university's Log-In page. Here "https://secure-access.morgan.edu/guest/ngnauth.php?" is a fixed string. "&mac=f8:b1:56:b3:7a:0a&_browser=1" is the MAC address of our machine and browser number which is the default browser.
5. **Lines 17-20:** Fill out the form on university's Log-In page using your username and password.
6. **Lines 21-22:** Click the box to accept the terms.
7. **Lines 23-24:** Click on the Accept button to Log-In.
8. **Lines 25-27:** Wait for the webpage to load to university's homepage then close the browser.
9. **Line 28:** Kill python.exe to stop the process cleanly.

```python
from selenium import webdriver
from selenium.webdriver.common.keys import Keys
import time
import os

temp = os.popen("ping 8.8.8.8").read()
test_connection1 = temp.find('PING: transmit failed. General
↪  failure.')
test_connection2 = temp.find('Request timed out.')

if (test_connection1 < 0 and test_connection2 < 0):
    print("Already connected. Exiting!!!")
else:
    driver = webdriver.Chrome()

    ↪  driver.get("https://secure-access.morgan.edu/guest/ngnauth.php?
                &mac=f8:b1:56:b3:7a:0a&_browser=1")

    elem1 =
    ↪  driver.find_element_by_id('ID_formcd662c6a_weblogin_user')
    elem1.send_keys("Peker.Milas")
    elem2 =
    ↪  driver.find_element_by_id('ID_formcd662c6a_weblogin_password')
    elem2.send_keys("Onder1981")
    elem3 =
    ↪  driver.find_element_by_id('ID_formcd662c6a_weblogin_visitor_accept_terms')
    elem3.click()
    elem4 =
    ↪  driver.find_element_by_id('ID_formcd662c6a_weblogin_submit')
    elem4.click()
    time.sleep(10)

    driver.close()
os.system("taskkill /F /im python.exe")
```

1.3 Implementation-Automatization

As I have indicated above, the provided Python script is meant to execute once. Therefore, to first sense a loss in wired connection and invoke the Selenium for automatic Log-In, we use Windows' **Task Scheduler**. This is done as following;

1. Click on the Windows search bar and type Task Scheduler. This will pop a widow above Windows logo to let you run Task Scheduler which is shown in Figure7

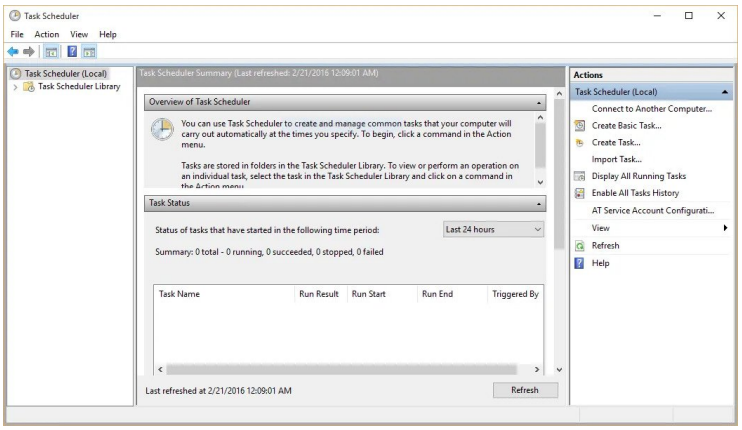

Figure 7: Windows10 Task Scheduler.

2. Click on the "Action→Create Task", as shown in Figure 8

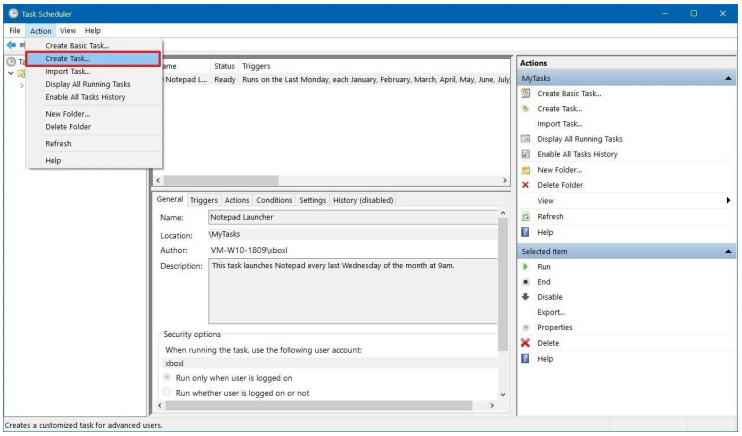

Figure 8: Create a task in Task Scheduler.

3. In popped-up window, on General tab, as shown in Figure9, give a unique name to your task and describe it for future reference.

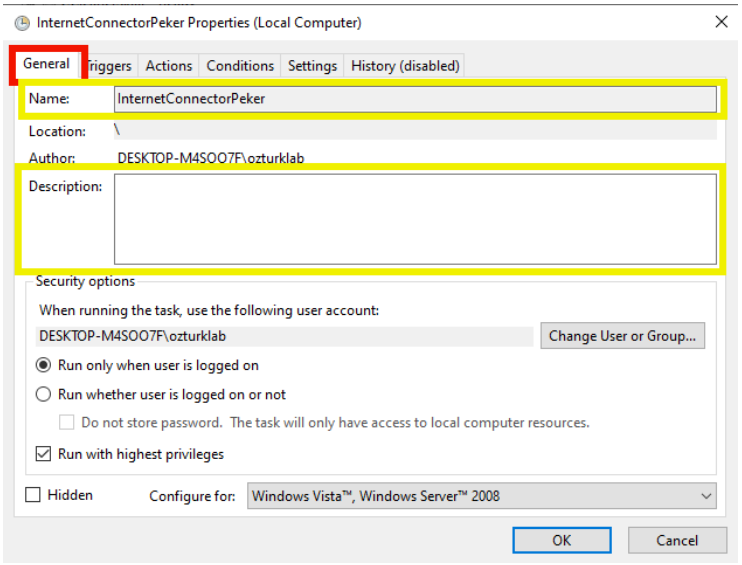

Figure 9: Name and Description fields under General tab.

4. On the Triggers tab, as shown in Figure10, click on New. In popped-up window, set the Start time and date. Also make your task runs for One time. Let the task repeat at specified times (here every 30 minutes) and keep it run indefinitely.

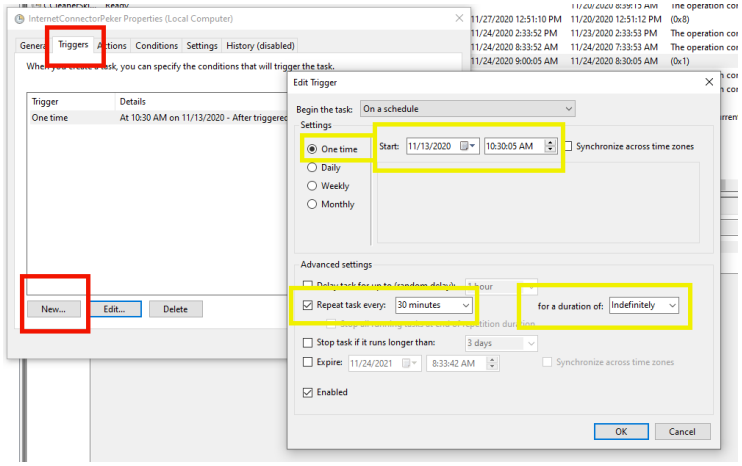

Figure 10: Timer and repetition related fields under Triggers tab.

5. On the Actions tab, as shown in Figure11, click on New. In popped-up window, make Action Start a program. In Programs/script field, add the

location of your python executable (python.exe). In Arguments field, add your python script's name. In Start in (Optional) field, add the location of your python script.

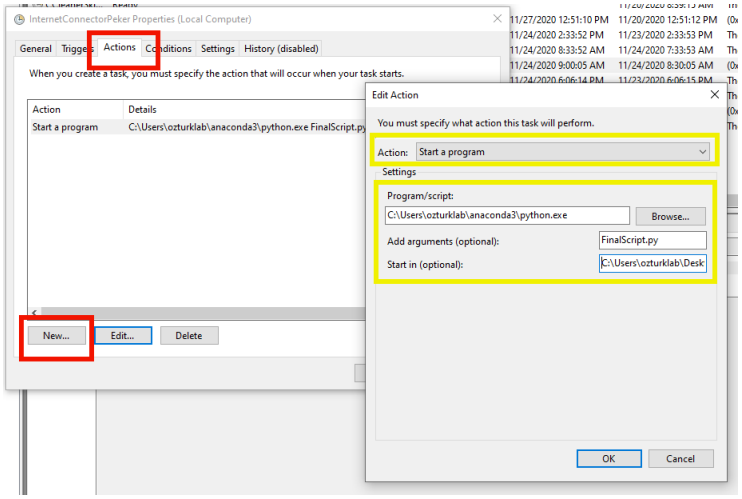

Figure 11: Script and arguments related fields under Triggers tab.

6. On the Conditions tab, as shown in Figure12, most of the fields are related with hardware such as whether you have a laptop which can run on battery or not. Therefore you can simply keep it as shown in Figure 12

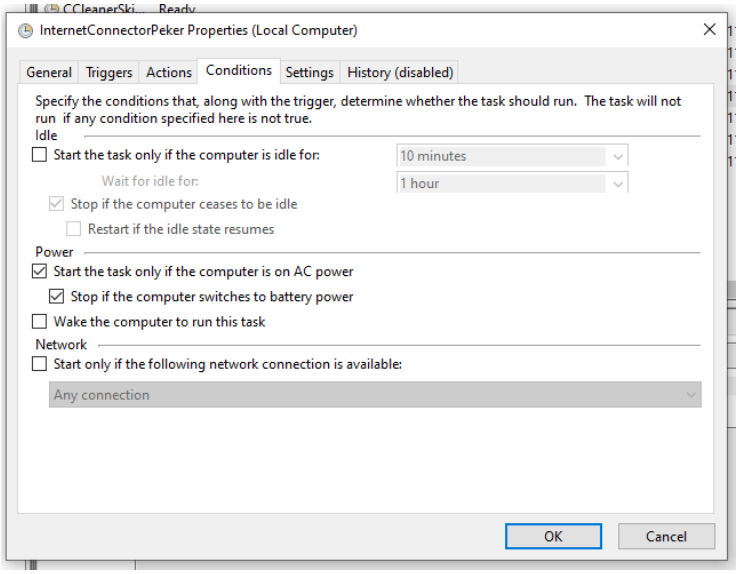

Figure 12: Already prepared Conditions tab.

7. On the Setting tab, as shown in Figure13, you can again set relevant fields as shown in Figure13.

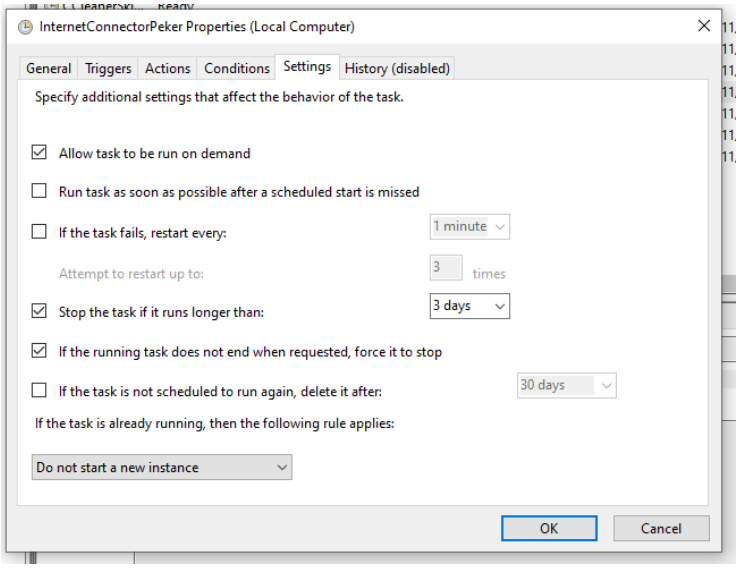

Figure 13: Already prepared Settings tab.

8. Once you are done with setting up your scheduled task, save it and close

its window. This will add your new task to default windows tasks as shown in Figure14. Right-click on your task and run it for the first time. After that point, it can run by itself.

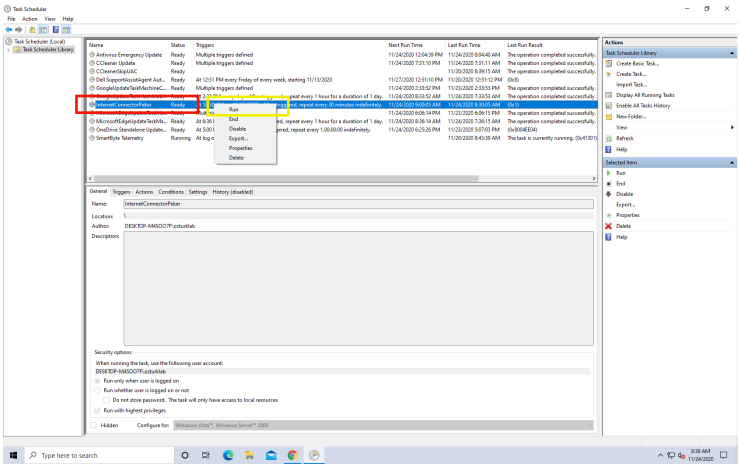

Figure 14: First run of the new task.

2 Photoluminescence (PL) Setup

Photoluminescence is the emission of light by the excited molecules and/or molecule clusters. Here we are intended to observe photon emission from the defects within crystalline structures such as diamond and cubic boron nitride. Given that these materials known to be opaque in visible light range, we are building a **reflecting light microscope**[refhere]. Although wide-field excitation (similar to epifluorescence [refhere]) based emission measurements from an ensemble of defects have their own importance in characterization, we pursue investigating properties of individual defects/ emitters. This requires our reflecting light microscope to be a **confocal system**.

2.1 Reflecting Light Microscopes Overview

On typical an upright or inverted microscope, illumination light comes from a direction relative to sample surface, while the collection or imaging is done in the opposite direction. Therefore in terms of optical components, illumination and imaging sides differ in those systems. For example in illumination side a special lens called **Condenser** focuses the light on to the sample while in the imaging side another special lens called **Objective** lens collects the light coming from the sample. These special lenses and the typical upright and inverted microscope configurations are shown in Figure15.

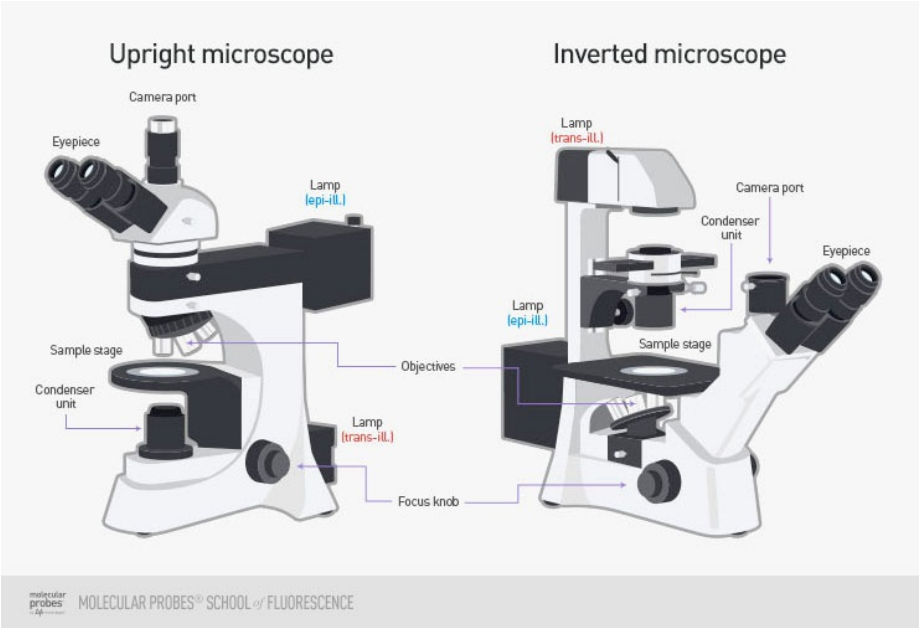

Figure 15: A typical upright (**left**) and an inverted (**right**) microscope structures indicating the positions of special lenses (**Condenser and objective**).

On the other hand, in a reflecting microscope illumination and imaging are done through the same optical path or by same optics. As it is shown in Figure16, shared optical path allows one (1) additional port for imaging and/or illuminating the sample on a typical reflecting microscope. This port is very frequently assigned to white-light (**bright-field**) imaging through a camera, so it is called the camera port. As a result any additional illumination and/or imaging on top of the default system, require a custom design optical system with the ability to finely couple with the existing microscope.

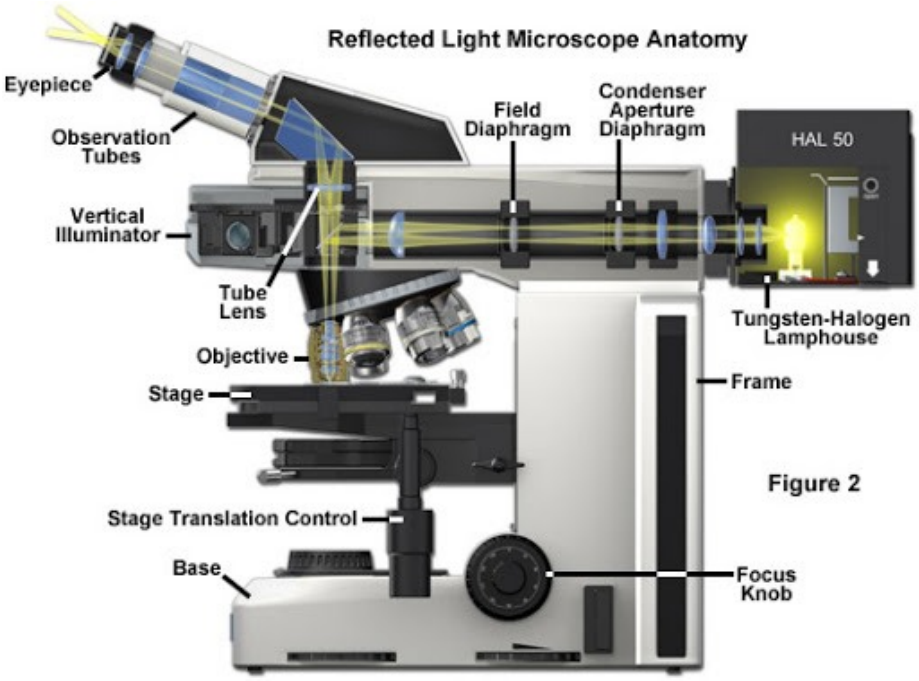

Figure 16: Schematics of a typical reflecting microscope with light path and internal optical components.

2.2 Confocal Systems Overview

A **confocal system** is an imaging system that utilizes a spatial filter (or simply a pinhole) before the imaging component for strictly matching the illumination volume with the observation volume as it is shown in Figure17. Because of very likely chromatic aberrations [3] through the imaging components, usually illumination is done by a single color light source such as a laser. The pinhole suppresses the out-of-excitation volume light dramatically, so it increases the resolution of imaging system, Figure18.

[3] `https://www.microscopyu.com/tutorials/chromatic`

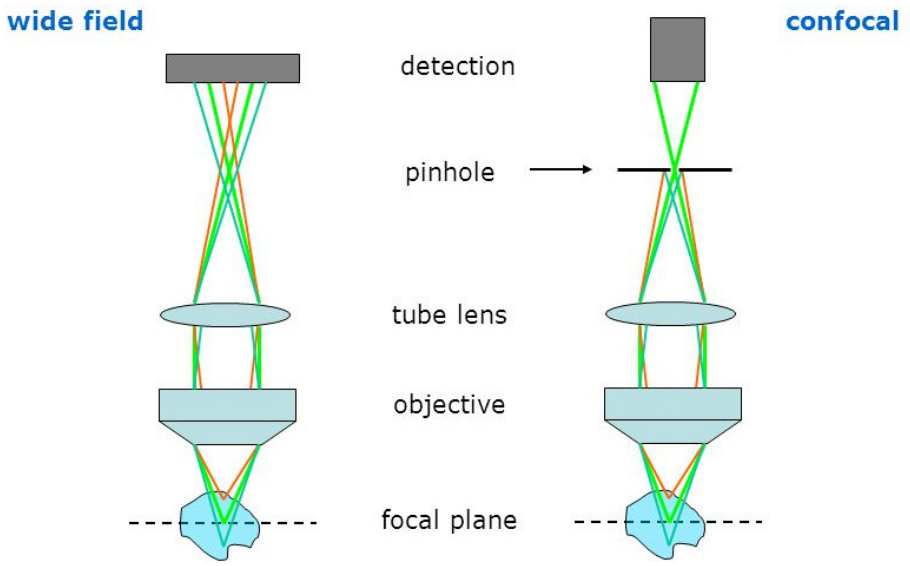

Figure 17: Simplified structure of a typical widefield (**left**) and a confocal (**right**) imaging system with spatial filter. ADD REF-FOOTNOTE HERE!!!

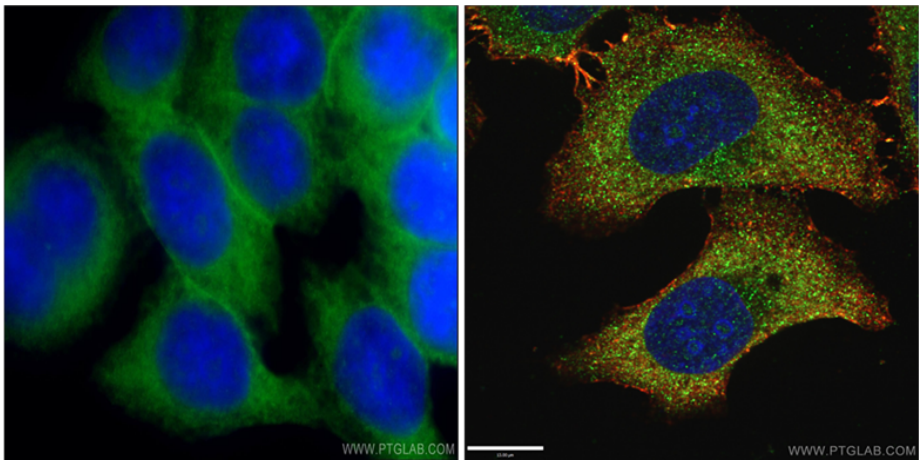

Figure 18: Imaging of HeLa cells with a typical widefield (**left**) and with a confocal (**right**) imaging system. ADD REF-FOOTNOTE HERE!!!

Microscope manufacturers very frequently incorporate additional openings (called **microscope ports**) on sides, front and even at the bottom of microscope bodies which allows the operator to couple additional imaging and illumination

systems into the microscope. Although this is a very common practice in the case of transmission microscopes (upright and inverted as well), reflecting microscopes usually don't have these additional ports, Figure16. Because of missing ports, turning a reflecting microscope is considerably harder into a confocal system compared to transmission microscopes.

2.3 Confocal Reflecting Light Microscope for PL Measurements

Our goal is to observe emission from excited lattice-defects in solids like boron nitride(BN) and diamond[REF HERE!!!]. These materials are not particularly transparent to visible light[4]. Moreover defect formation (particularly in the case of BN) is induced by ion bombardment, so neither the density nor spatial distribution of defects are known prior to a PL measurement. The lack of information on defect positions introduces the requirement of active search for defects within the sample volume, so a confocal imaging system with the capability of scanning. To overcome the refractive index issue and satisfying the confocality in imaging, we built a confocal reflecting light microscope as shown in Figure19.

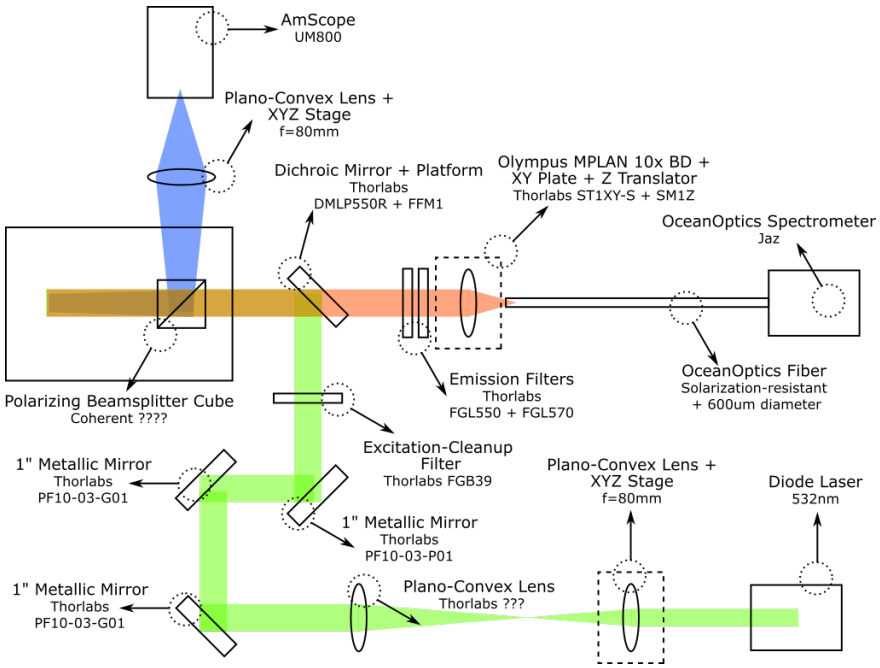

Figure 19: Our homemade confocal reflecting microscope for PL measurements.

Although the purpose of individual parts can not be explained accurately

[4] https://en.wikipedia.org/wiki/Boron_nitride

within the structure, it is a common practice to divide the whole system into 4 essential zones for educational purposes as in Figure20.

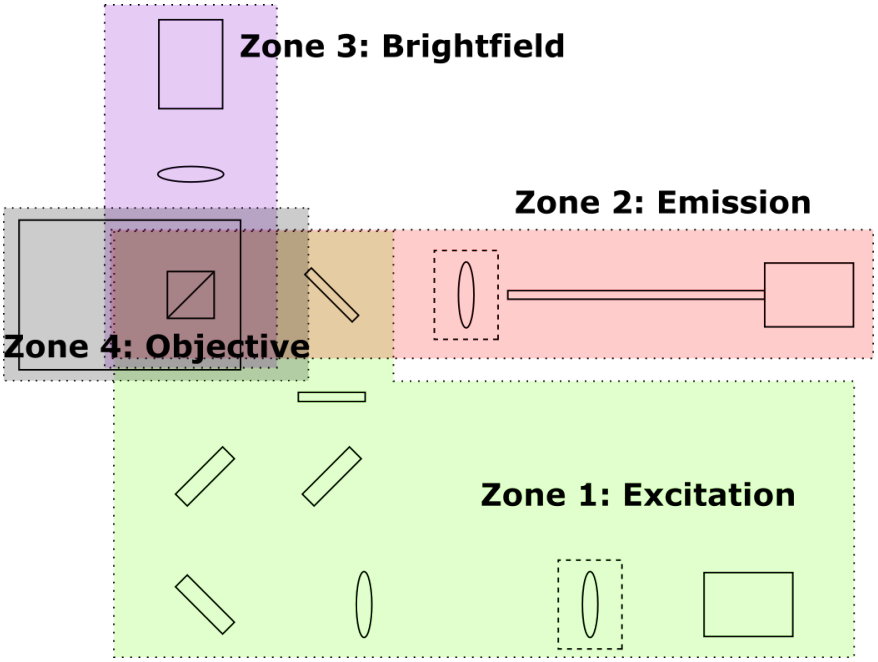

Figure 20: Zones in PL microscope.