Any variable listed here can be modified by including an entry in your dictionary that has a matching key. Some variables have useful defaults that are mentioned. It is only necessary to include a value in the dictionary you send if that value has changed, but it is important to keep in mind that changing one of these values might necessarily involve changing another. For example, if the ‘columns’ and ‘rows’ variables get changed, then the size of the ‘pattern’ and ‘s0\_thresholds’ variables must necessarily be changed to compensate. User errors do to consistency are likely to appear as internal server errors in the log feed (HTTP code 500). The variables are grouped according to the page you would make a PUT or POST request to in order to modify them. All arrays must be provided as lists and not numpy arrays because numpy arrays are not JSON serializable.

‘/python\_settings’

‘doRearrangement’: Boolean value that determines whether the server will calculate assignments and pass instructions along to the microcontrollers. Defaults to True

‘columns’: The number of columns in the array. Defaults to 11.

‘rows’: The number of rows in the array. Defaults to 11.

‘top’: The y coordinate for the top-left atom site in the region of rearrangement. Defaults to 0.

‘left’: The x coordinate for the top-left atom site in the region of rearrangement. Defaults to 0.

‘width’: The width (x axis) of the region of rearrangement. Defaults to same as ‘columns’.

‘height’: The height (y axis) of the region of rearrangement. Defaults to same as ‘rows’.

‘gaussian\_roi\_params’: Tuple of values that serve as arguments to the function that generates ROI Masks for whole array.

‘cutoff’: Turn any value generated by the gaussian roi code into a 1 if it is above this value and 0 if it is below when generating masks. Defaults to 0.769.

‘pattern’: array of numbers encoding the desired array configuration. 1 means we want the site filled. 0 means we want the site empty. Any other value means we don’t care. Defaults to “don’t care” for all sites.

‘s0\_thresholds’: array of values that serve as cutoffs for determining loading for whole array. Defaults to all 0s.

An example request could be made as follows:

import requests

d = {}

d[‘columns’] = 3

d[‘rows’] = 3

d[‘top’] = 0

d[‘left’] = 0

d[‘width’] = 3

d[‘height’] = 3

#The 80,80 in the first element of the tuple is the dimensions of the camera data received by LABVIEW

#These values are arbitrary

d[‘gaussian\_row\_params’] = ((80,80),

14.988,20.09,-0.052,-0.040,5.132,0.041,1.414,1.328,1.998,908078.1,6.611)

d[‘pattern’] = [ 1, 1, 1, 1, 0, 1, 1,1,1] #Want center dot empty but rest filled

d[‘s0\_thresholds]’ = [5, 5, 5, 5, 5, 5, 5, 5, 5] #Arbitrary thresholds in the example

requests.put('http://10.0.0.2:5000/python\_settings',json= d)

‘/arduino\_settings’

‘frequency\_increment’: The difference between frequency values written to the AODs (or max resolution if you request something too precise). Defaults to 0.01 Mhz/write

‘jump\_time’: The time in microseconds to wait in between successive writes to the AODs. Defaults to 100.

‘laser\_ramp\_on\_time’: The amount of time the laser takes to turn on or off in microseconds. Defaults to 300.

‘fitfrequencies’: List of tuples formatted as [(site #, (xf, yf)] used to fit lines and determine all site frequencies.

‘forcefrequencies’: List of tuples formatted as [(site #, (xf, yf)] used to override the fitting solution when desired.

Example:

import requests

d={}

d[‘frequency\_increment’] = 0.01

d[‘jump\_time’] = 100

d[‘laser\_ramp\_on\_time’] = 300

d[‘fitfrequencies’] = [ ( 0, (130,130) ), (1, (131,130) ), ( 2, (132,130) ), ( 3, (131,130) ), ( 4, (131,131) ) ]

d[‘forcefrequencies’] = [ (8, (140,140) )] #for whatever reason this site is an outlier and we force it

requests.put('http://10.0.0.2/arduino\_settings',json= d)