Import the required libraries

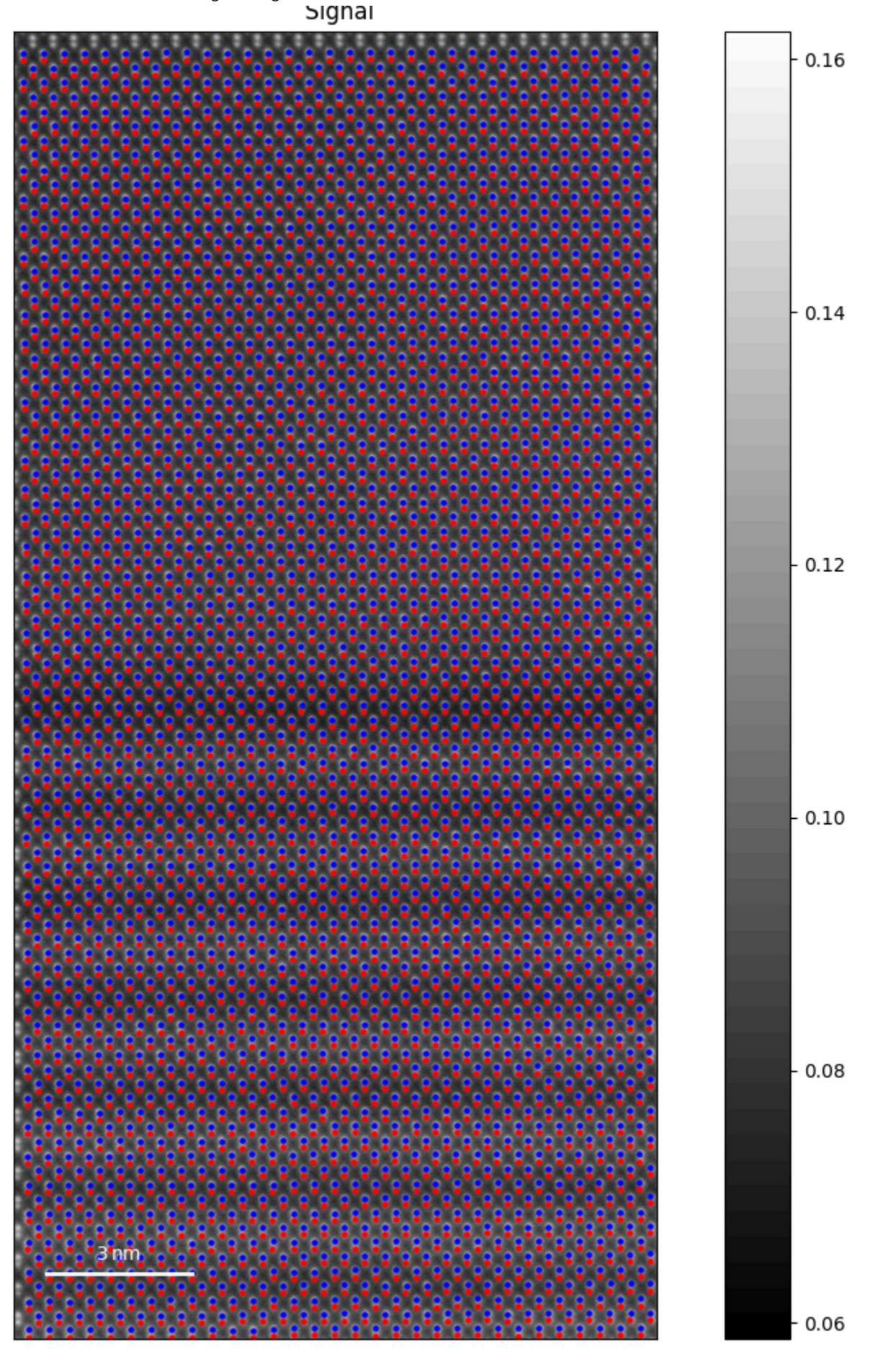
In [1]:
```
%matplotlib widget
import hyperspy.api as hs
import numpy as np
import matplotlib.pyplot as plt
import atomap.api as am
import tkinter as tk
from tkinter.filedialog import askdirectory
import os
hs.preferences.GUIs.warn_if_guis_are_missing = False
hs.preferences.save()
plt.rcParams['figure.figsize'] = (8,8)
```

Intensity Maps

- Loading the atom lattice:

In [2]:
```
root = tk.Tk()
root.attributes('-topmost',True)
root.iconify()
file_path = askdirectory(parent=root)
root.destroy()
atom_lattice = am.load_atom_lattice_from_hdf5(file_path+'\\data.hdf5',construct_zone_axes=False)
sublattice_A=atom_lattice.sublattice_list[1]
sublattice_B=atom_lattice.sublattice_list[0]
atom_lattice.units=atom_lattice.sublattice_list[0].units
atom_lattice.pixel_size=atom_lattice.sublattice_list[0].pixel_size
atom_lattice.plot(markersize=5)
ax = plt.gca()
fig=plt.gcf()
fig.set_size_inches((12,10))
ax.get_xaxis().set_visible(False)
ax.get_yaxis().set_visible(False)
```

Figure Signal



> `text`; selected processed image:

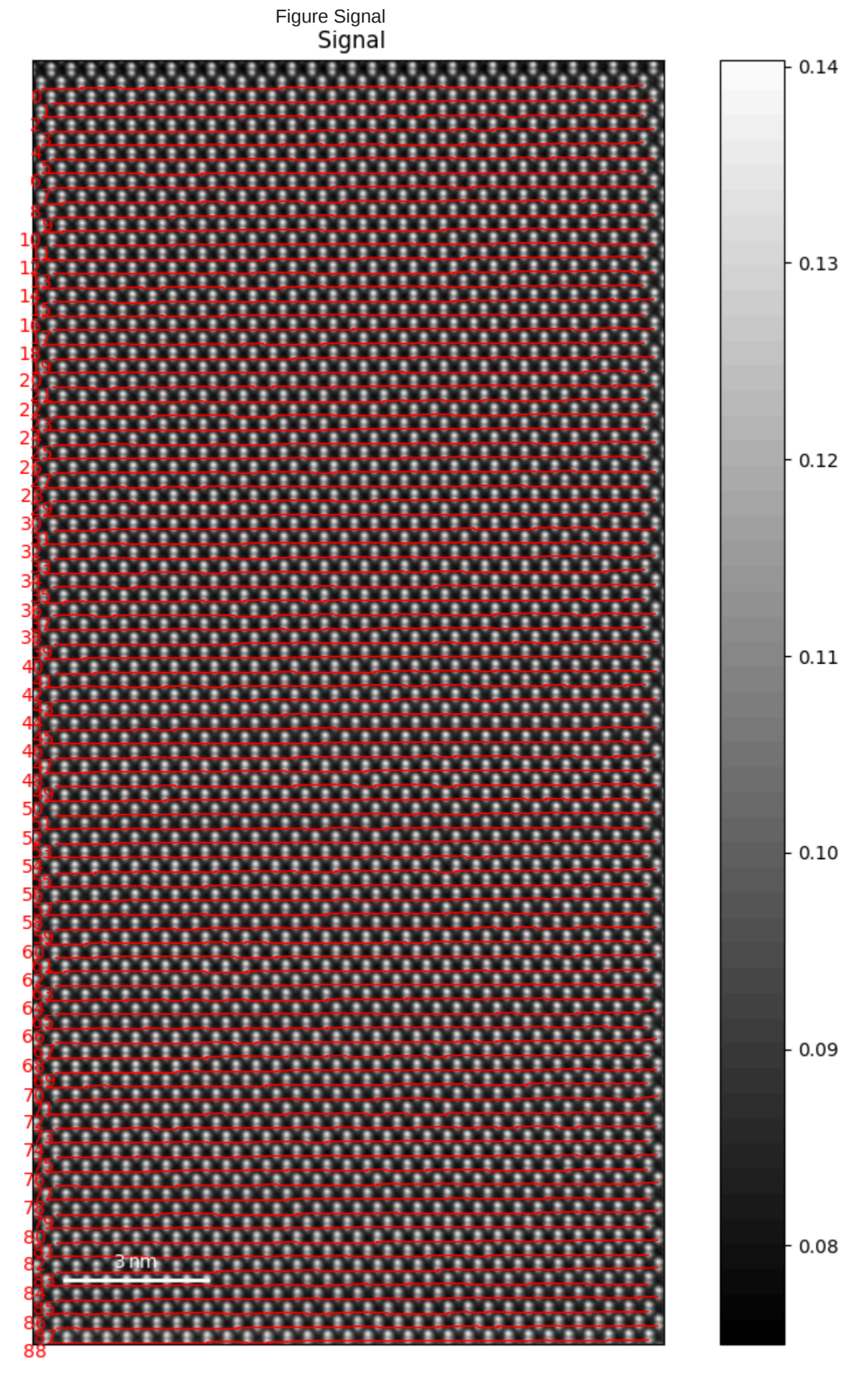
- `high_pass_original_imag` for high-pass only.
- `band_pass_original_imag` for band-pass only.
- `high_pass_pca_imag` for PCA with high-pass.
- `band_pass_pca_imag` for PCA with band-pass.
- `high_pass_nlm_imag` for NL-means with high-pass.
- `band_pass_nlm_imag` for NL-means with band-pass.
- `high_pass_rl_imag` for RL deconvolution with band-pass.
- `band_pass_rl_imag` for RL deconvolution with band-pass.

In [3]:
```
text='band_pass_pca_imag'

if  text!='':
    sublattice_A.original_image=np.load(file_path+'\\'+text+'.npy')
sublattice_A.find_nearest_neighbors(nearest_neighbors=15)
sublattice_A._pixel_separation = sublattice_A._get_pixel_separation()
sublattice_A._make_translation_symmetry()
if ((0,0) in sublattice_A.zones_axis_average_distances):
    index=sublattice_A.zones_axis_average_distances.index((0,0))
    sublattice_A.zones_axis_average_distances.remove(sublattice_A.zones_axis_average_distances[index])
    sublattice_A.zones_axis_average_distances_names.remove(sublattice_A.zones_axis_average_distances_names[index])
sublattice_A._generate_all_atom_plane_list(atom_plane_tolerance=1.5)
sublattice_A._sort_atom_planes_by_zone_vector()
sublattice_A._remove_bad_zone_vectors()
```

> `direction`: zone axis index.

In [4]:
```
direction=2

zone_vector = sublattice_A.zones_axis_average_distances[direction]
atom_planes = sublattice_A.atom_planes_by_zone_vector[zone_vector]
zone_axis = sublattice_A.get_atom_planes_on_image(atom_planes)
zone_axis.plot()
ax = plt.gca()
fig=plt.gcf()
fig.set_size_inches((10,10))
ax.get_xaxis().set_visible(False)
ax.get_yaxis().set_visible(False)
plt.show()
```

Figure Signal



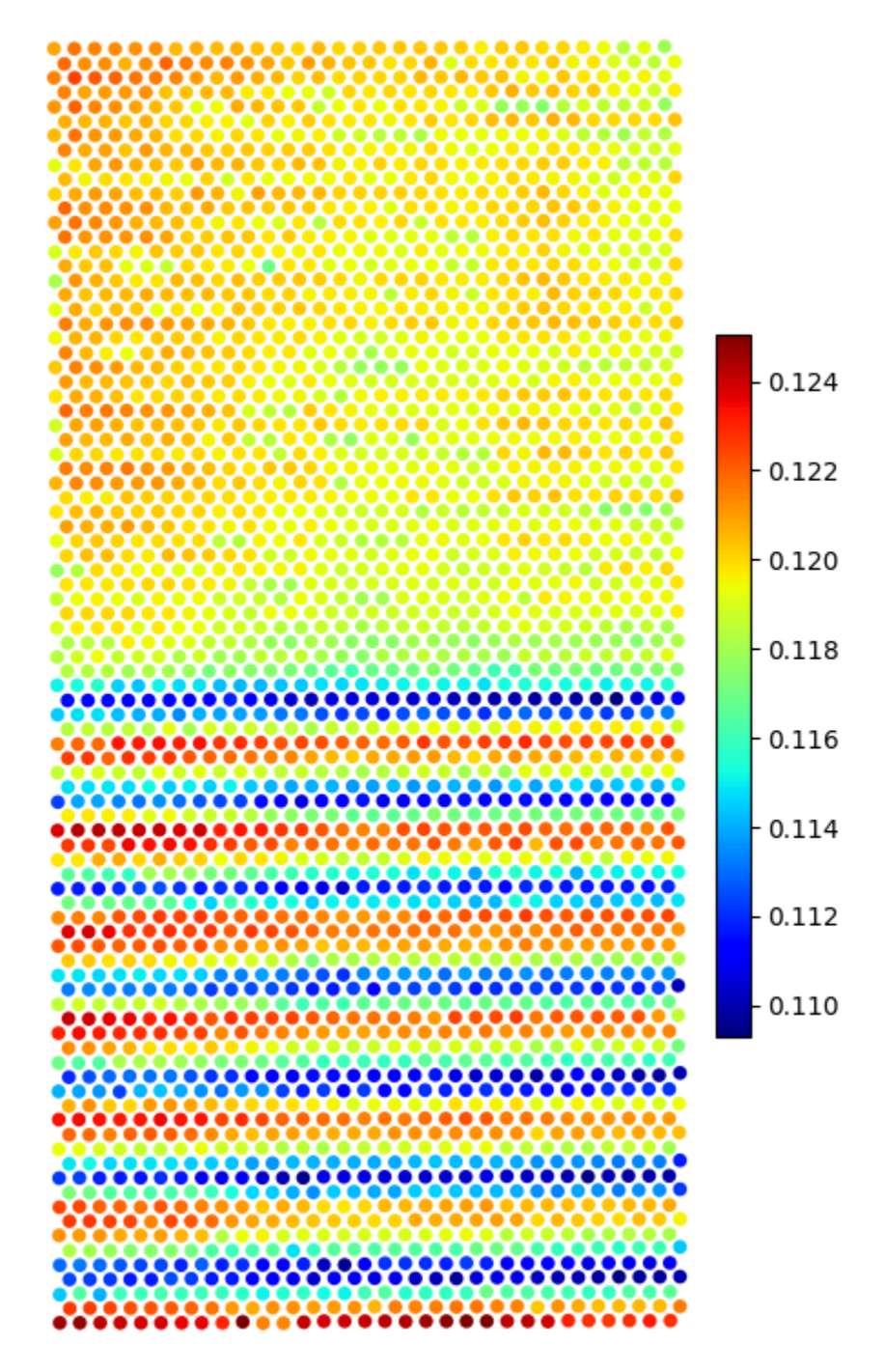
- Intensity map of the sublattice A:

In [5]:
```
sublattice_A.find_sublattice_intensity_from_masked_image(sublattice_A.original_image,radius=7)
zone_axis_A = sublattice_A.zones_axis_average_distances[direction]
atom_plane_list_A = sublattice_A.atom_planes_by_zone_vector[zone_axis_A]
intensity_A=[]
x_values=[]
y_values=[]
for i in range(0,len(atom_plane_list_A)):
    atomplane=atom_plane_list_A[i]
    plane_intensity=[]
    x_values_plane=[]
    y_values_plane=[]
    for j in range(0, len(atomplane.atom_list)):
        atom=atomplane.atom_list[j]
        x_pos,y_pos=atom.get_pixel_position()
        intensity=atom.intensity_mask
        plane_intensity.append(intensity)
        x_values_plane.append(x_pos)
        y_values_plane.append(y_pos)
    intensity_A.append(plane_intensity)
    x_values.append(x_values_plane)
    y_values.append(y_values_plane)

intensity_A_array = np.zeros([len(intensity_A),len(max(intensity_A,key = lambda x: len(x)))])
intensity_A_array[:] = np.nan
for i,j in enumerate(intensity_A):
    intensity_A_array[i][0:len(j)] = j

x_values_array = np.zeros([len(x_values),len(max(x_values,key = lambda x: len(x)))])
x_values_array[:] = np.nan
for i,j in enumerate(x_values):
    x_values_array[i][0:len(j)] = j

y_values_array = np.zeros([len(y_values),len(max(y_values,key = lambda x: len(x)))])
y_values_array[:] = np.nan
for i,j in enumerate(y_values):
    y_values_array[i][0:len(j)] = j

intensity_A=np.stack((intensity_A_array,x_values_array,y_values_array),axis=2)
```

```
Finding intensity:   0%|          | 0/2759 [00:00<?, ?it/s]
```

In [6]:
```
plt.figure(figsize=(6,8))
plt.scatter(intensity_A[:,:,1],intensity_A[:,:,2],s=20,c=intensity_A[:,:,0],cmap='jet')
plt.colorbar(shrink=0.5,pad=-0.18)
plt.axis('scaled')
plt.axis('off')
ax=plt.gca()
ax.set_ylim(ax.get_ylim()[::-1])
ax.xaxis.tick_top()
ax.yaxis.tick_left()
plt.tight_layout()
plt.show()
```

Figure



In [7]:
```
plt.savefig(file_path+'\\im_A_'+text+'.png',dpi=300,transparent=True,bbox_inches='tight')
np.save(file_path+'\\im_A_'+text+'.npy',intensity_A)
```

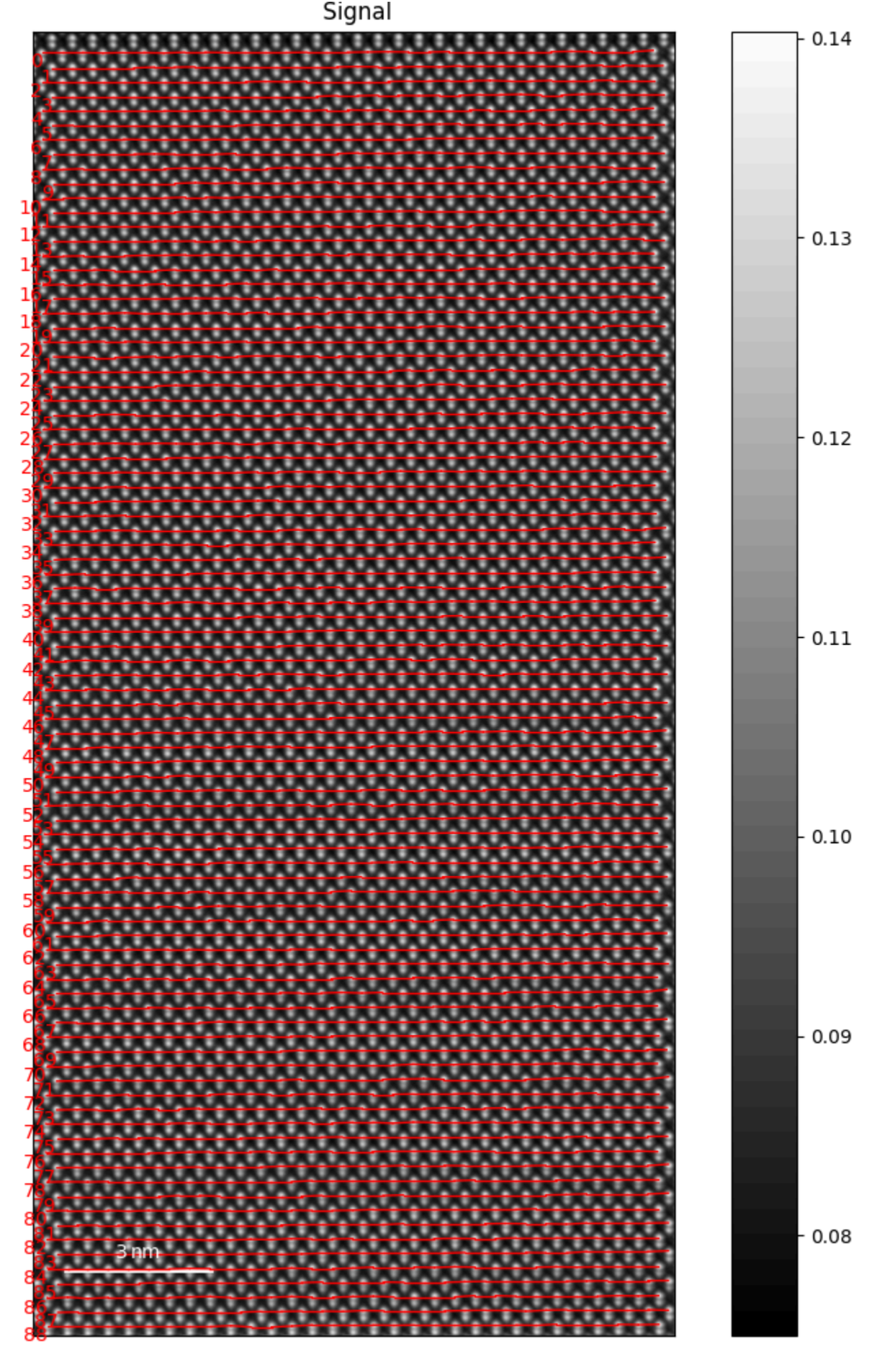
- Same process for sublattice B:

In [8]:
```
if  text!='':
    sublattice_B.original_image=np.load(file_path+'\\'+text+'.npy')
sublattice_B.find_nearest_neighbors(nearest_neighbors=15)
sublattice_B._pixel_separation = sublattice_B._get_pixel_separation()
sublattice_B._make_translation_symmetry()
if ((0,0) in sublattice_B.zones_axis_average_distances):
    index=sublattice_B.zones_axis_average_distances.index((0,0))
    sublattice_B.zones_axis_average_distances.remove(sublattice_B.zones_axis_average_distances[index])
    sublattice_B.zones_axis_average_distances_names.remove(sublattice_B.zones_axis_average_distances_names[index])
sublattice_B._generate_all_atom_plane_list(atom_plane_tolerance=1.5)
sublattice_B._sort_atom_planes_by_zone_vector()
sublattice_B._remove_bad_zone_vectors()
```

> `direction`: zone axis index.

In [9]:
```
direction=2

zone_vector = sublattice_B.zones_axis_average_distances[direction]
atom_planes = sublattice_B.atom_planes_by_zone_vector[zone_vector]
zone_axis = sublattice_B.get_atom_planes_on_image(atom_planes)
zone_axis.plot()
ax = plt.gca()
fig=plt.gcf()
fig.set_size_inches((10,10))
ax.get_xaxis().set_visible(False)
ax.get_yaxis().set_visible(False)
plt.show()
```

Figure Signal



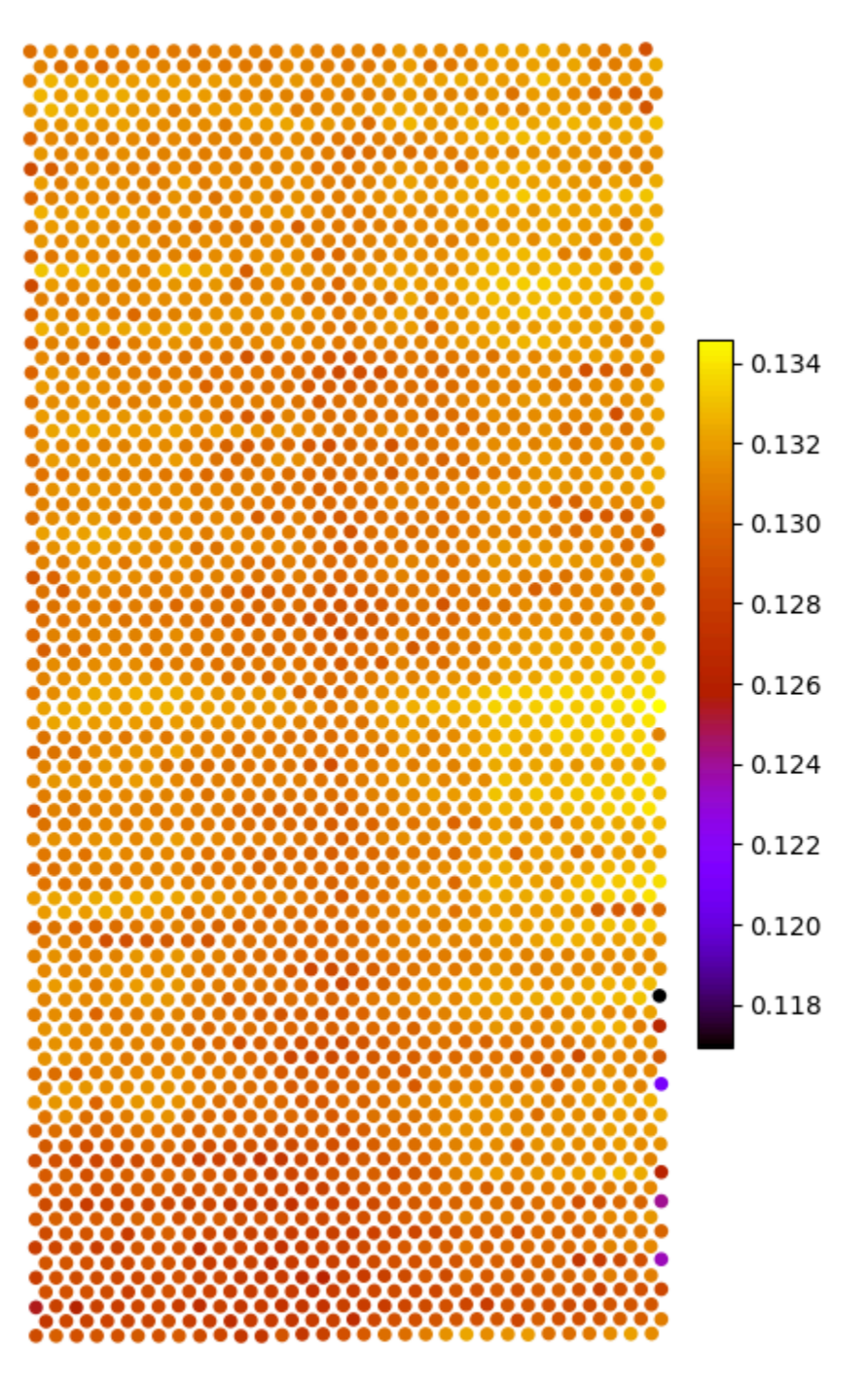
- Intensity map of the sublattice B:

In [10]:
```
sublattice_B.find_sublattice_intensity_from_masked_image(sublattice_B.original_image,radius=5)
zone_axis_B = sublattice_B.zones_axis_average_distances[direction]
atom_plane_list_B = sublattice_B.atom_planes_by_zone_vector[zone_axis_B]
intensity_B=[]
x_values=[]
y_values=[]
for i in range(0,len(atom_plane_list_B)):
    atomplane=atom_plane_list_B[i]
    plane_intensity=[]
    x_values_plane=[]
    y_values_plane=[]
    for j in range(0, len(atomplane.atom_list)):
        atom=atomplane.atom_list[j]
        x_pos,y_pos=atom.get_pixel_position()
        intensity=atom.intensity_mask
        plane_intensity.append(intensity)
        x_values_plane.append(x_pos)
        y_values_plane.append(y_pos)
    intensity_B.append(plane_intensity)
    x_values.append(x_values_plane)
    y_values.append(y_values_plane)

intensity_B_array = np.zeros([len(intensity_B),len(max(intensity_B,key = lambda x: len(x)))])
intensity_B_array[:] = np.nan
for i,j in enumerate(intensity_B):
    intensity_B_array[i][0:len(j)] = j

x_values_array = np.zeros([len(x_values),len(max(x_values,key = lambda x: len(x)))])
x_values_array[:] = np.nan
for i,j in enumerate(x_values):
    x_values_array[i][0:len(j)] = j

y_values_array = np.zeros([len(y_values),len(max(y_values,key = lambda x: len(x)))])
y_values_array[:] = np.nan
for i,j in enumerate(y_values):
    y_values_array[i][0:len(j)] = j

intensity_B=np.stack((intensity_B_array,x_values_array,y_values_array),axis=2)
```

```
Finding intensity:   0%|          | 0/2759 [00:00<?, ?it/s]
```

In [11]:
```
plt.figure(figsize=(6,8))
plt.scatter(intensity_B[:,:,1],intensity_B[:,:,2],s=20,c=intensity_B[:,:,0],cmap='gnuplot')
plt.colorbar(shrink=0.5,pad=-0.18)
plt.axis('scaled')
plt.axis('off')
ax=plt.gca()
ax.xaxis.tick_top()
ax.yaxis.tick_left()
plt.tight_layout()
ax.set_ylim(ax.get_ylim()[::-1])
plt.show()
```

Figure

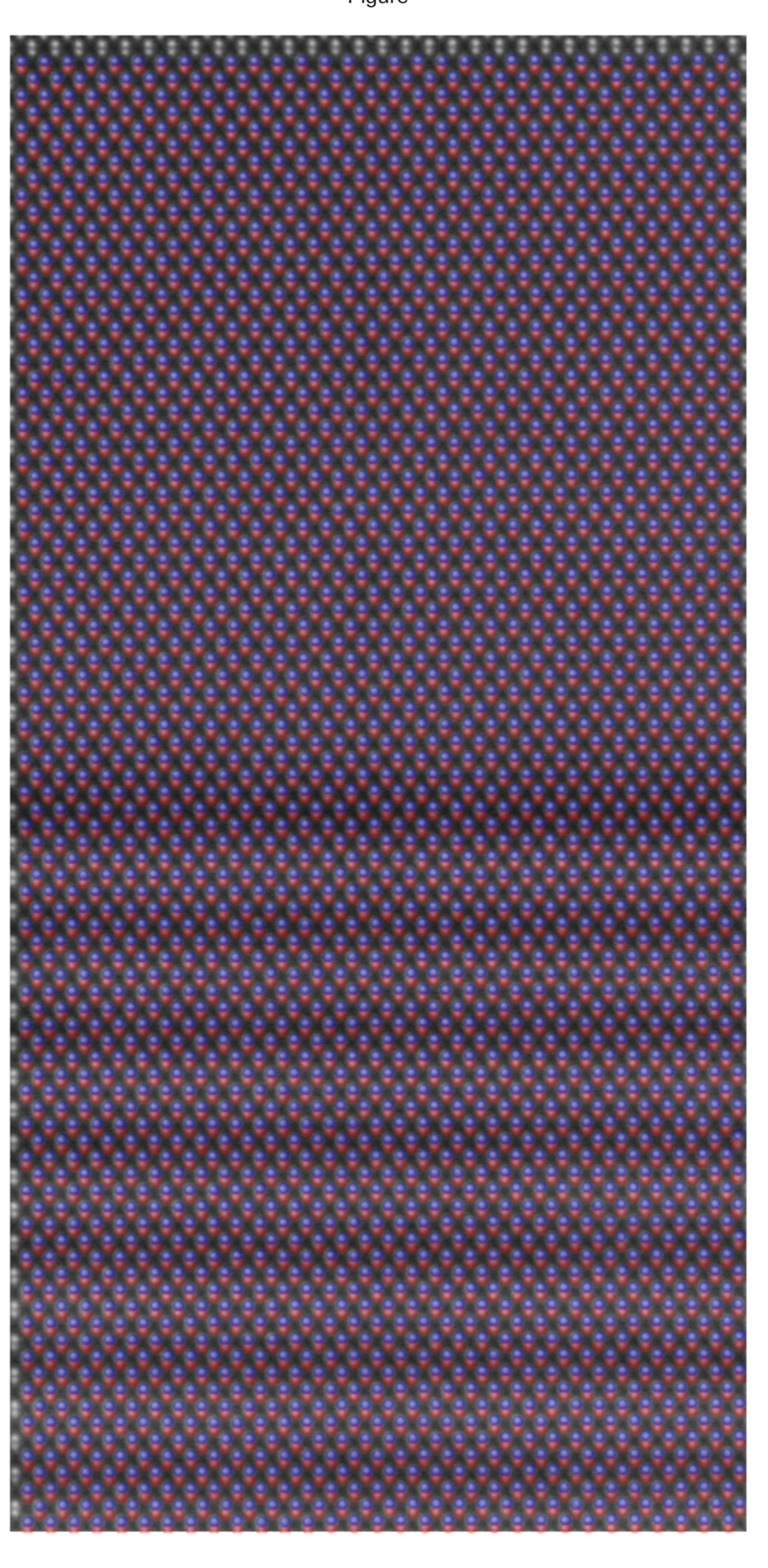


In [12]:
```
plt.savefig(file_path+'\\im_B_'+text+'.png',dpi=300,transparent=True,bbox_inches='tight')
np.save(file_path+'\\im_B_'+text+'.npy',intensity_B)
```

- Atom positions:

In [13]:
```
plt.figure(figsize=(12,10))
plt.imshow(atom_lattice.original_image,cmap='gray')
plt.scatter(intensity_A[:,:,1],intensity_A[:,:,2],s=20,c='red',alpha=0.25)
plt.scatter(intensity_B[:,:,1],intensity_B[:,:,2],s=20,c='blue',alpha=0.25)
plt.axis('scaled')
plt.axis('off')
ax=plt.gca()
ax.xaxis.tick_top()
ax.yaxis.tick_left()
plt.tight_layout()
```

Figure

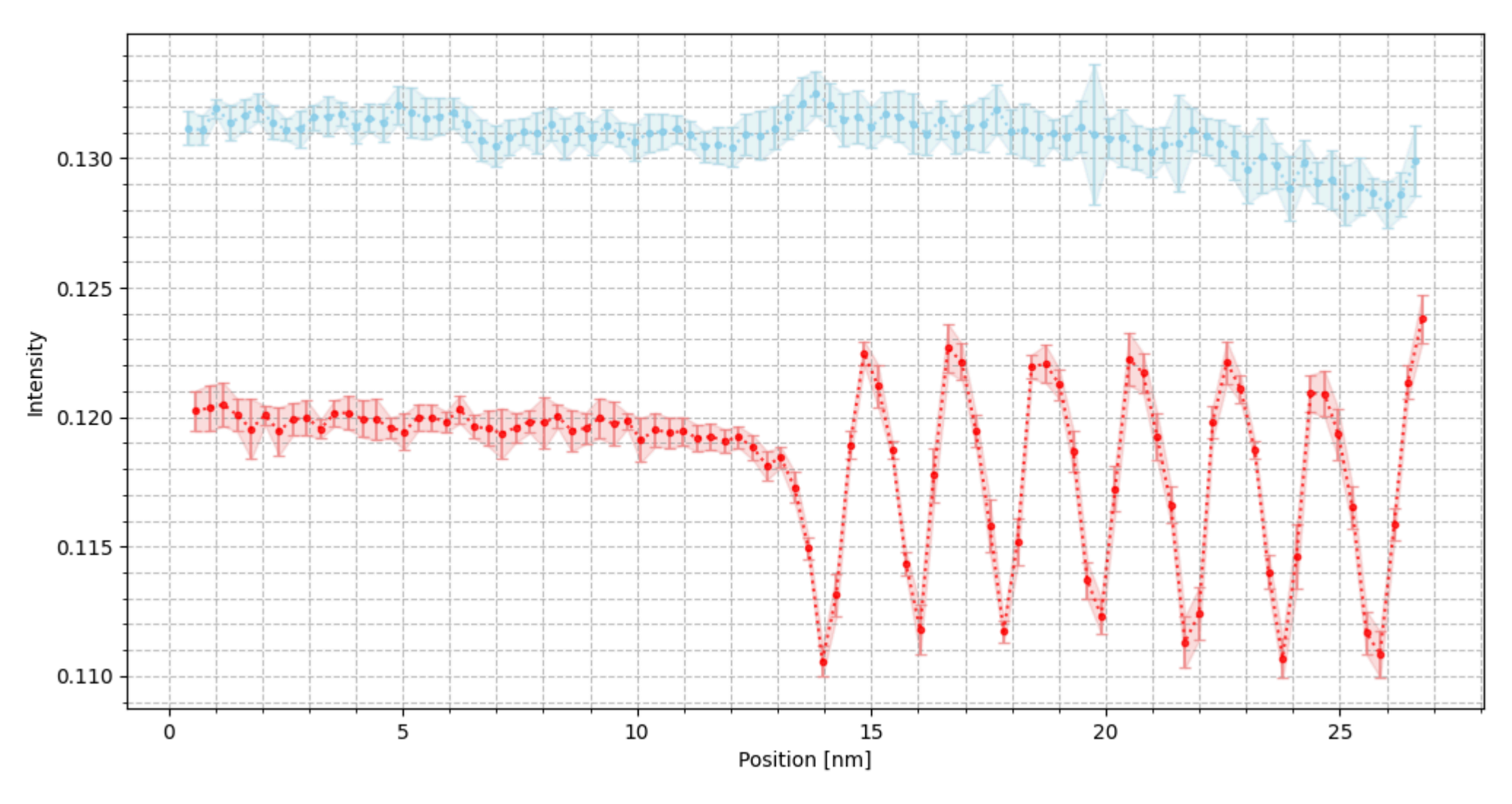


In [14]:
```
plt.savefig(file_path+'\\intensitymap_all.png',dpi=300,transparent=True,bbox_inches='tight')
```

Composition profile

In [15]:
```
avg_intensity=np.nanmean(intensity_A[:,:,0],axis=1)
avg_std=np.nanstd(intensity_A[:,:,0],axis=1)
avg_axis=np.nanmean(intensity_A[:,:,2],axis=1)*atom_lattice.pixel_size
avg_intensity1=np.nanmean(intensity_B[:,:,0],axis=1)
avg_std1=np.nanstd(intensity_B[:,:,0],axis=1)
avg_axis1=np.nanmean(intensity_B[:,:,2],axis=1)*atom_lattice.pixel_size
plt.figure(figsize=(12,6))
plt.errorbar(avg_axis,avg_intensity, yerr=avg_std, ecolor='lightcoral',marker='.',fmt=':', capsize=3,alpha=0.75, mec='red',mfc='red',color='red' )
plt.fill_between(avg_axis,avg_intensity-avg_std,avg_intensity+avg_std, alpha=.25, color='lightcoral')
plt.errorbar(avg_axis1,avg_intensity1, yerr=avg_std1, ecolor='lightblue',marker='.',fmt=':', capsize=3,alpha=0.75, mec='skyblue',mfc='skyblue',color='skyblue' )
plt.fill_between(avg_axis1,avg_intensity1-avg_std1,avg_intensity1+avg_std1, alpha=.25, color='lightblue')
plt.xlabel('Position [nm]')
plt.ylabel('Intensity')
plt.minorticks_on()
plt.grid(which='both',linestyle='--',color='gray',alpha=0.5)
ax=plt.gca()
plt.show()
```

Figure



In [16]:
```
textA='A_'+str(round(np.mean(avg_intensity),7))+'+-'+str(round(np.std(avg_intensity),7))+'^'+str(round(np.mean(avg_std),7))
textB='B_'+str(round(np.mean(avg_intensity1),7))+'+-'+str(round(np.std(avg_intensity1),7))+'^'+str(round(np.mean(avg_std1),7))
plt.savefig(file_path+'\\ip_'+text+'.png',dpi=300,transparent=True,bbox_inches='tight')
os.rename(file_path+'\\im_A_'+text+'.png',file_path+'\\A_'+text+textA+'.png')
os.rename(file_path+'\\im_B_'+text+'.png',file_path+'\\B_'+text+textB+'.png')
```