Physics 391

Patrick Selep

Laboratory 1

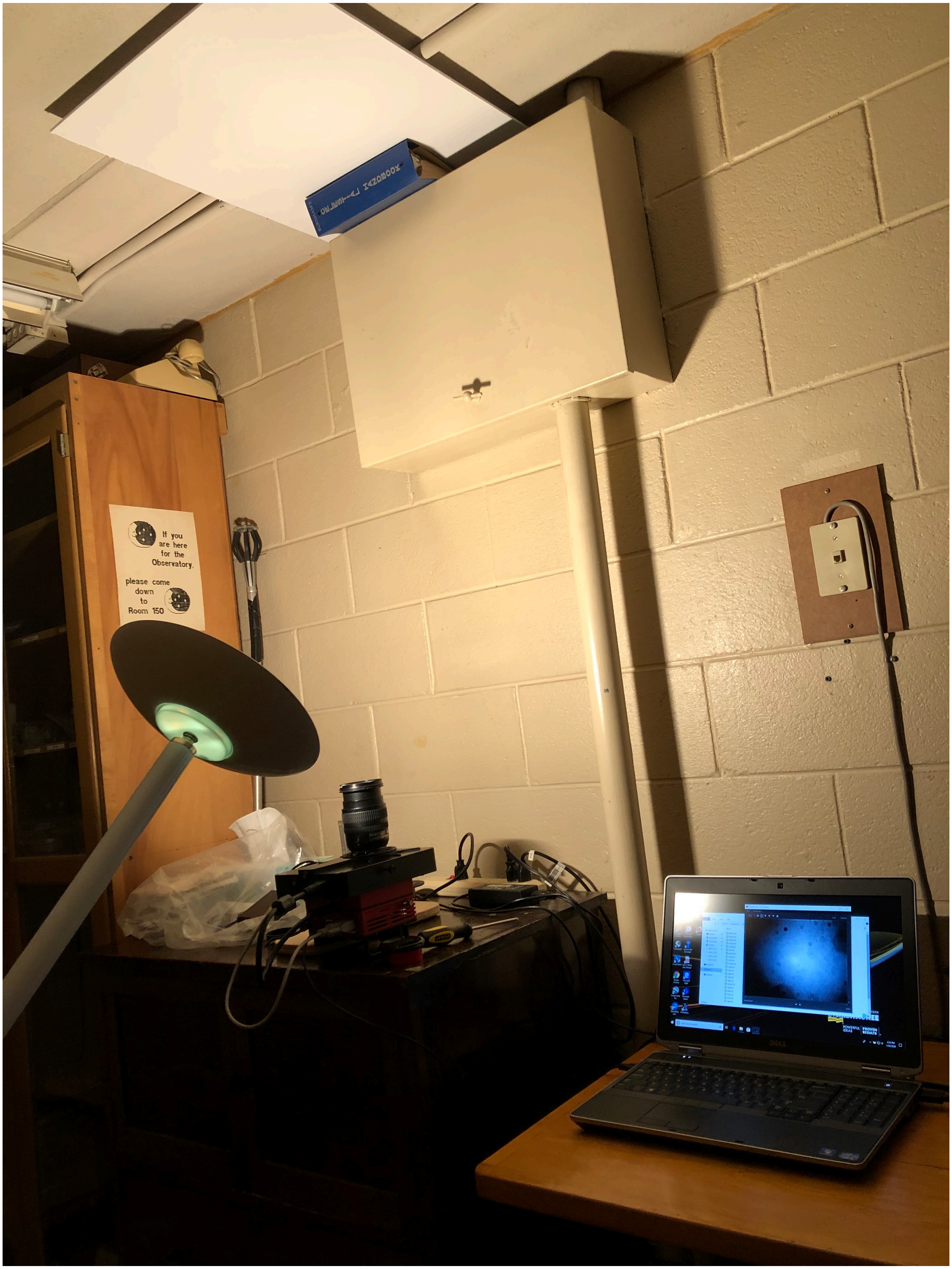
CCD Performance Characteristics

This laboratory covered assessing several performance characteristics of the CCD. Specifically, the Gain, Read Noise, Dark Current and Linearity were measured.

Experimental Apparatus

The SBIG STT 8300 was tested on a bench through a camera lens. A white board was mounted to the ceiling above the camera. The board illuminated with a halogen lamp with a dimmer control. Linearity was tested at varying exposure times up to saturation. Intensity was also varied as was the lighting arrangement. Bias and dark images were taken with the lense cap on. Bias images were taken with an exposure time of zero. Dark images were taken with varying exposure times up to 16 minutes.

Here is a picture of the arrangement of the equipment.



Performance Characteristics

Performance Characteristics

Gain and Read Noise were calculated from a pair of matched flat and bias images. The formulas used to determine them are shown below:

$$\text{Gain} = \frac{(\bar{F_1} + \bar{F_2}) - (\bar{B_1} + \bar{B_2})}{\sigma^2_{F_1 - F_2} - \sigma^2_{B_1 - B_2}}$$

$$\text{Read Noise} = \frac{\text{Gain} \cdot \sigma_{B_1 - B_2}}{\sqrt{2}}$$

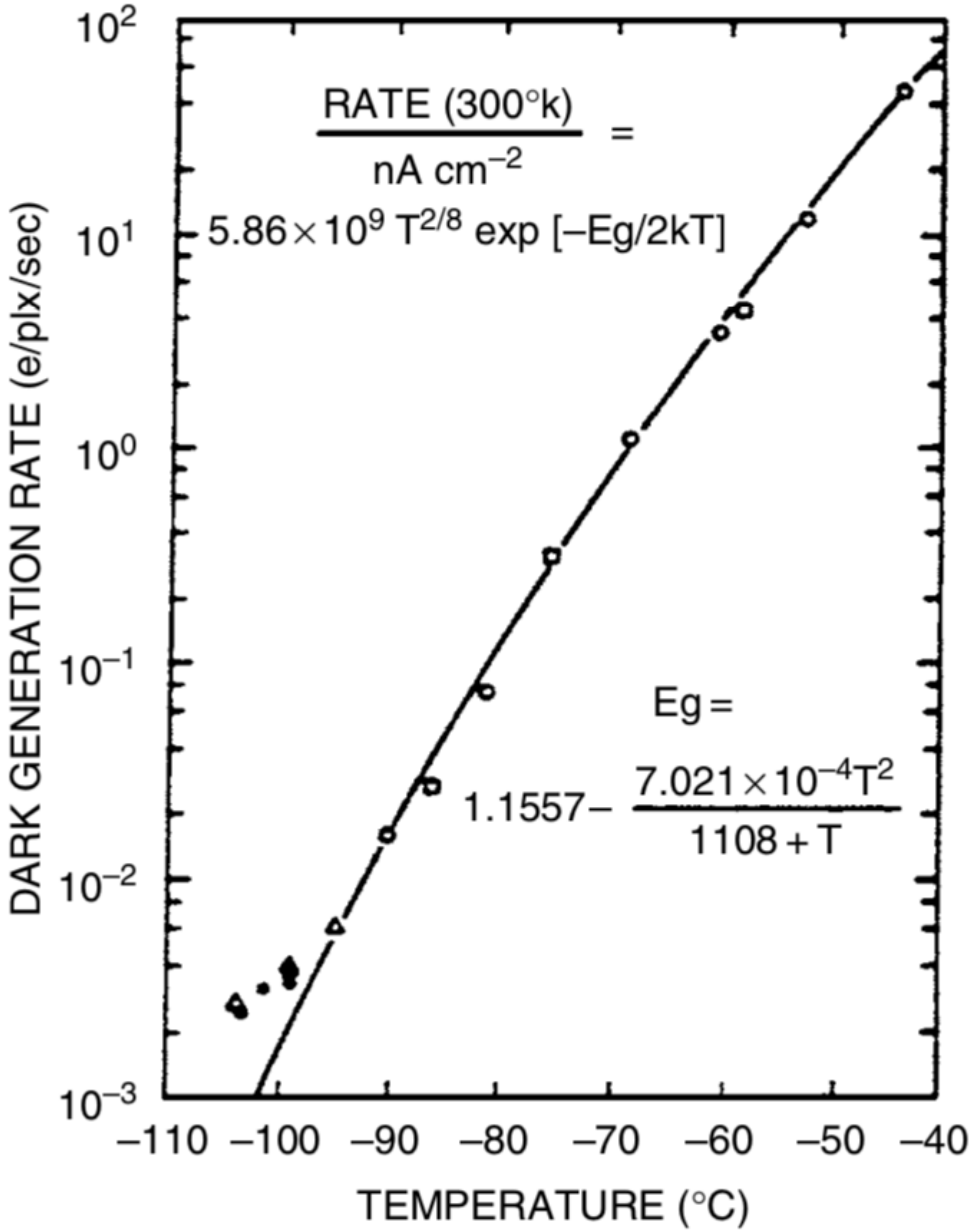
The results were as follows:

The Gain was found to be 0.379 e-/ADU compared to the published 0.37 e-/adu.

The Read Noise was found to be 10.663 e- rms compared to the published <10 e- rms.

The Dark Current was found to be 0.328 e-/pixel/sec at -1 C compared to 0.02 e-/pixel/se

Dark Current is highly dependent on temperature. From the Handbook of CCD Astronomy "Typical values for properly cooled devices range from 2 electrons per second per pixel down to very low levels of approximately 0.04 electrons per second for each pixel."



The Linearity was found to be linear with and R = .999974 up to saturation.

The residuals of this plot show that counts are below predictions at high counts.

Manufacturer's Specifications

Model	STT-1603ME	STT-3200ME	STT-8300M
CCD	Kodak KAF-1603ME	Kodak KAF-3200ME	Kodak KAF-8300
Pixel Array	1536 x 1024 pixels @ 9u	2184 x 1510 pixels @ 6.8u	3326 x 2504 pixels
CCD Size	13.8 x 9.2 mm	14.85 x 10.26 mm	17.96 x 13.52 mm
Total Pixels	1.6 million	3.2 million	8.3 million
Full Well Capacity	100,000 e-	55,000 e-	25,500 e-
Dark Current	0.1e-/pixel/sec @ -20C.	0.06e-/pixel/sec at -20C.	0.02e-/pixel/sec at -15C.
Antiblooming	NABG only	NABG Only	1000X
Shutter	Mechanical, Even-illumination	Mechanical, Even-illumination	Mechanical, Even-illumination
Exposure	0.12 to 3600 seconds, 10ms	0.12 to 3600 seconds, 10ms	0.12 to 3600 seconds, 10ms
A/D Converter	16 bits	16 bits	16 bits
Gain	2.3e-/ADU	1.3e-/ADU	0.37e-/ADU
Read Noise	< 15e- rms	10e- rms	< 10e- rms
Binning Modes	1x1, 2x2, 3x3, 9x9, x n	1x1, 2x2, 3x3, 9x9, x n	1x1, 2x2, 3x3, 9x9, x n
Digitization Rate	10 Megapixels per second	8.33 Megapixels per second	10 Megapixels per second
Full Frame Download	< 1 second	< 1 second	< 1 second
Max Cooling Delta	-55C with air only	-55C with air only	-55C with air only
Temp. Regulation	±0.1°C	±0.1°C	±0.1°C
Power	12VDC at 3.5 amps	12VDC at 3.5 amps	12VDC at 3.5 amps
Interface	USB 2.0 and Ethernet	USB 2.0 and Ethernet	USB 2.0 and Ethernet
Computer Compatibility	Windows 32 / 64 bit	Windows 32 / 64 bit	Windows 32 / 64 bit
	Mac OSX, 3rd party Linux	Mac OSX, 3rd party Linux	Mac OSX, 3rd party Linux
Camera Body Size	4.9 x 4.9 x 2.9 in.	4.9 x 4.9 x 2.9 in.	4.9 x 4.9 x 2.9 in.
	124 x 124 x 74mm	124 x 124 x 74mm	124 x 124 x 74mm
Mounting	T-Thread, 2" nosepiece	T-Thread, 2" nosepiece	T-Thread, 2" nosepiece
Weight	2.7 pounds / 1.2kg	2.7 pounds / 1.2kg	2.7 pounds / 1.2kg
Backfocus	0.69 inches / 17.5 mm	0.69 inches / 17.5 mm	0.69 inches / 17.5 mm

In [1]:

```
from astropy.io import fits
from os import walk
from matplotlib import pyplot as plt

import numpy as np
from scipy import stats

```

In [2]:

```
#with fits.open('2020-01-23/65sFF.fit') as f:
#    f.info()
#    scidata = f[0].data.copy()
```

In [3]:
```
#print(scidata.shape)
#print(scidata.dtype)
```

In [4]:
```
#f[0].header
```

In [5]:
```
#f[0].header['EXPTIME']
```

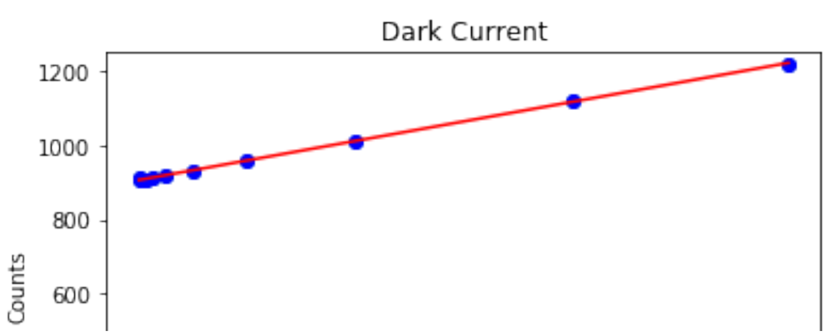
In [6]:
```
# Find GAIN

# Get the data from an HDU.
F1data = fits.getdata('2020-01-23/Gain/2sFF1.fit').astype(float)
#print(f"min: {F1data.min()}, max: {F1data.max()}, mean: {F1data.mea
Flat1Mean = F1data.mean()
Flat1Std = F1data.std()
F2data = fits.getdata('2020-01-23/Gain/2sFF2.fit').astype(float)
#print("min: {}, max: {}, mean: {:.3f}, std: {:.3f}, Flat2".format(F
Flat2Mean = F2data.mean()
Flat2Std = F2data.std()
Z1data = fits.getdata('2020-01-23/Gain/00sBS1.fit').astype(float)
#print("min: {}, max: {}, mean: {:.3f}, std: {:.3f}, Dark1".format(Z
Zero1Mean = Z1data.mean()
Zero1Std = Z1data.std()
Z2data = fits.getdata('2020-01-23/Gain/00sBS2.fit').astype(float)
#print("min: {}, max: {}, mean: {:.3f}, std: {:.3f}, Dark2".format(Z
Zero2Mean = Z2data.mean()
Zero2Std = Z2data.std()

flatdif = (F1data - F2data)

flatstd = flatdif.std()
#print("flatstd: {:.3f}".format(flatstd))
zerodif = Z1data - Z2data
zerostd = zerodif.std()
#print("zerostd: {:.3f}".format(zerostd))
gain = ((Flat1Mean + Flat2Mean) - (Zero1Mean + Zero2Mean)) / ((flats


readnoise = gain * zerostd / np.sqrt(2)
print("gain: {:.3f}, readnoise: {:.3f}".format(gain, readnoise))

```

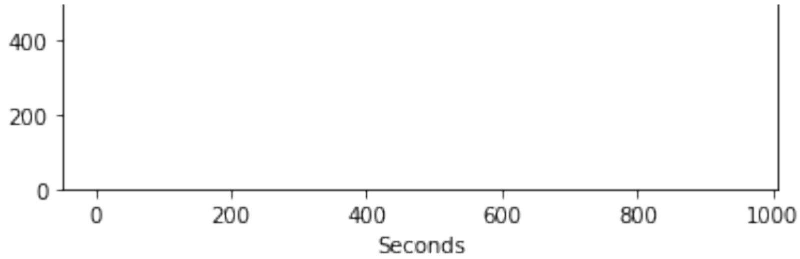
```
gain: 0.379, readnoise: 10.663
```

Dark Current

In [13]:
```
# READ Files
```

```
from pathlib import Path
Means = []
Expos = []

basepath = Path('2020-02-04/')
files_in_basepath = basepath.iterdir()
for item in files_in_basepath:
    if item.is_file():
        with fits.open(item) as f:
#             f.info()
#         print(item.name, (f[0].header['EXPTIME']))
            data = fits.getdata(basepath/item.name)
            Means.append(data.mean())
            Expos.append(f[0].header['EXPTIME'])
#             print("mean: {}, exp: {}".format(data.mean(), (f[0].hea
#             plt.scatter((f[0].header['EXPTIME']),data.mean())

plt.scatter(Expos,Means)
plt.title('Dark Current')
plt.xlabel('Seconds')
plt.ylabel('Counts')
plt.ylim(0,1250)

#

gradient, intercept, r_value, p_value, std_err = stats.linregress(Ex
mn=np.min(Expos)
mx=np.max(Expos)
x1=np.linspace(mn,mx,500)
y1=gradient*x1+intercept
plt.plot(Expos,Means,'ob')
plt.plot(x1,y1,'-r')
plt.savefig('Lab1_DarkCurrent.png')
plt.show()

print("slope: {:.6f}, R: {:.6f}, P: {:.6f}, Std_err: {:.6f}".format(

with fits.open('2020-02-04/640sDK.fit') as f:
    print("Dark Current: {:.3f}, at Temp: {:.3f}".format(gradient,f[
```






```
slope: 0.328064, R: 0.999794, P: 0.000000, Std_err: 0.002221
Dark Current: 0.328, at Temp: -0.888
```

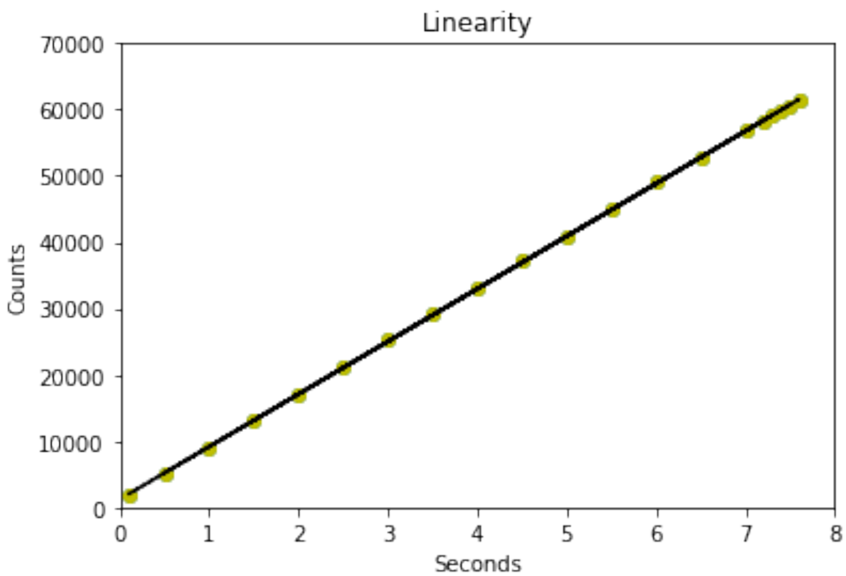
Linearity

In [14]:

```
# READ Files

from pathlib import Path
Means = []
Expos = []

basepath = Path('2020-01-23/Fast/')
files_in_basepath = basepath.iterdir()
for item in files_in_basepath:
    if item.is_file():
        with fits.open(item) as f:
#             f.info()
#        print(item.name, (f[0].header['EXPTIME']))
            data = fits.getdata(basepath/item.name)
            Means.append(data.mean())
            Expos.append(f[0].header['EXPTIME'])
#             print("mean: {}, exp: {}".format(data.mean(), (f[0].hea
#             plt.scatter((f[0].header['EXPTIME']),data.mean())

plt.scatter(Expos,Means)
plt.title('Linearity')
plt.xlabel('Seconds')
plt.ylabel('Counts')
plt.ylim(0,70000)

#
gradient, intercept, r_value, p_value, std_err = stats.linregress(Ex
mn=np.min(Expos)
mx=np.max(Expos)
x1=np.linspace(mn,mx,500)
y1=gradient*x1+intercept

#
coef = np.polyfit(Expos,Means,1)
poly1d_fn = np.poly1d(coef)
```

```
# poly1d_fn is now a function which takes in x and returns an estima

plt.plot(Expos,Means, 'yo', Expos, poly1d_fn(Expos), '--k')
#plt.plot(x1,y1+1000,'-r')


plt.xlim(0, 8)
plt.ylim(0, 70000)
plt.savefig('Lab1_Linearity.png')
plt.show

print("slope: {:.6f}, R: {:.6f}, P: {:.6f}, Std_err: {:.6f}".format(


```

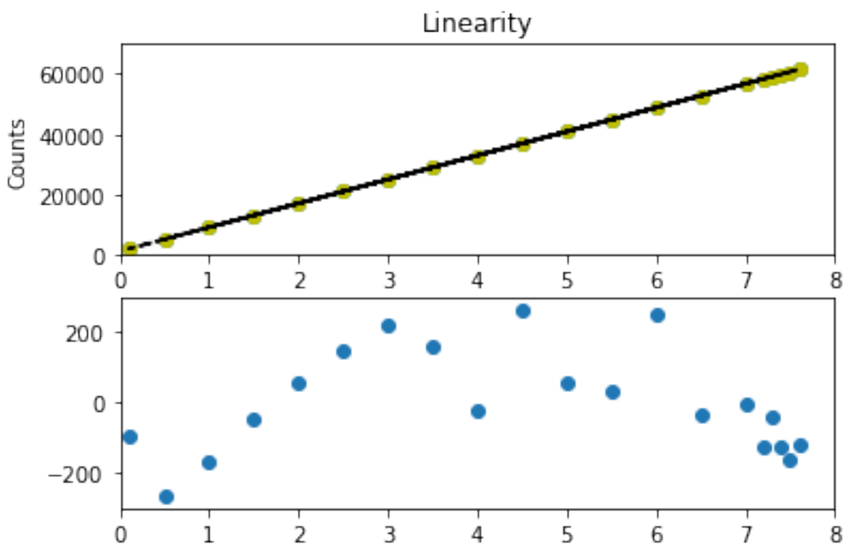
```
slope: 7934.225589, R: 0.999974, P: 0.000000, Std_err: 13.591658
```



In [9]:

```

#PLOT

plt.subplot(211)
plt.scatter(Expos,Means)
plt.title('Linearity')
plt.xlabel('Seconds')
plt.ylabel('Counts')
plt.ylim(0,70000)

coef = np.polyfit(Expos,Means,1)
poly1d_fn = np.poly1d(coef)
# poly1d_fn is now a function which takes in x and returns an estima

plt.plot(Expos,Means, 'yo', Expos, poly1d_fn(Expos), '--k')
plt.xlim(0, 8)
```

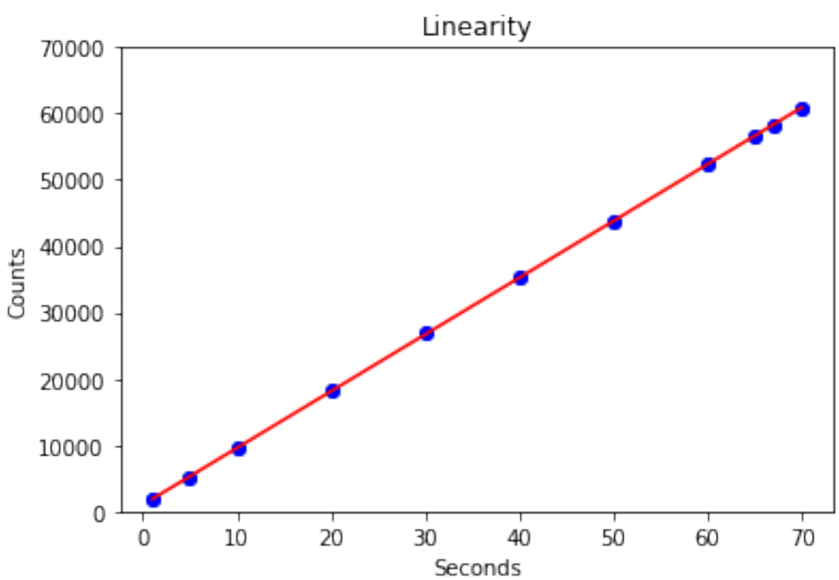
```
plt.ylim(0, 70000)
plt.show


#Residual plot

plt.subplot(212)
plt.scatter(Expos,Means-poly1d_fn(Expos))
#plt.scatter(x1,Means-y1)


plt.xlim(0, 8)
plt.ylim(-300, 300)
plt.savefig('Lab1_DC_Residuals.png')
plt.show()


```



In [16]:

```
# READ Files

from pathlib import Path
Means = []
Expos = []

basepath = Path('2020-01-23/Slow/')
files_in_basepath = basepath.iterdir()
for item in files_in_basepath:
    if item.is_file():
        with fits.open(item) as f:
#              f.info()
#          print(item.name, (f[0].header['EXPTIME']))
            data = fits.getdata(basepath/item.name)
            Means.append(data.mean())
            Expos.append(f[0].header['EXPTIME'])
```

```
#            print("mean: {}, exp: {}".format(data.mean(), (f[0].hea
#            plt.scatter((f[0].header['EXPTIME']),data.mean())

plt.scatter(Expos,Means)
plt.title('Linearity')
plt.xlabel('Seconds')
plt.ylabel('Counts')
plt.ylim(0,70000)

#


gradient, intercept, r_value, p_value, std_err = stats.linregress(Ex
mn=np.min(Expos)
mx=np.max(Expos)
x1=np.linspace(mn,mx,500)
y1=gradient*x1+intercept
plt.plot(Expos,Means,'ob')
plt.plot(x1,y1,'-r')
plt.savefig('Lab1_DarkCurSlow.png')
plt.show()

print("slope: {}, R: {}, P: {}, Std_err: {}".format(gradient,r_value
#
```



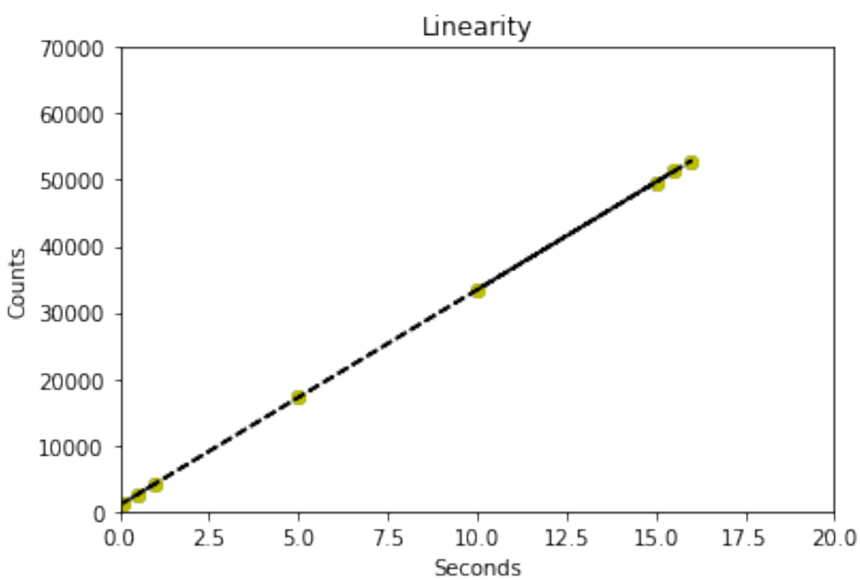

```
slope: 855.9002635086536, R: 0.9999802673596974, P: 3.942172701229881e
-21, Std_err: 1.7923215654218903
```

In [15]:

```
# READ Files

from pathlib import Path
Means = []
Expos = []
```

```
basepath = Path('2020-01-23/Med/')
files_in_basepath = basepath.iterdir()
for item in files_in_basepath:
    if item.is_file():
        with fits.open(item) as f:
#            f.info()
#         print(item.name, (f[0].header['EXPTIME']))
            data = fits.getdata(basepath/item.name)
            Means.append(data.mean())
            Expos.append(f[0].header['EXPTIME'])
#            print("mean: {}, exp: {}".format(data.mean(), (f[0].hea
#            plt.scatter((f[0].header['EXPTIME']),data.mean())

plt.scatter(Expos,Means)
plt.title('Linearity')
plt.xlabel('Seconds')
plt.ylabel('Counts')
plt.ylim(0,70000)

#
coef = np.polyfit(Expos,Means,1)
poly1d_fn = np.poly1d(coef)
# poly1d_fn is now a function which takes in x and returns an estima

plt.plot(Expos,Means, 'yo', Expos, poly1d_fn(Expos), '--k')
plt.xlim(0, 20)
plt.ylim(0, 70000)

#
plt.savefig('Lab1_DarkCurMed.png')
plt.show
```

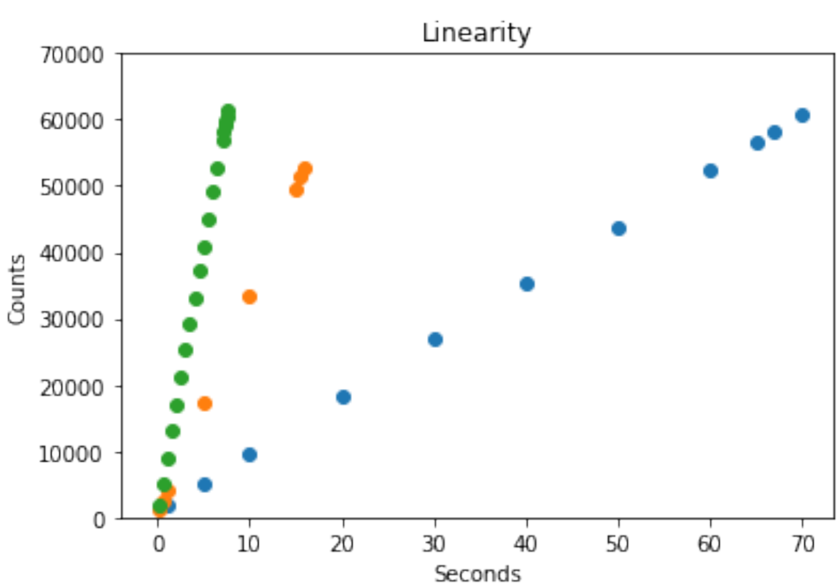
Out[15]: <function matplotlib.pyplot.show(*args, **kw)>



In [17]:
```
# READ Slow Files

from pathlib import Path
SMeans = []
SExpos = []

basepath = Path('2020-01-23/Slow/')
files_in_basepath = basepath.iterdir()
for item in files_in_basepath:
    if item.is_file():
        with fits.open(item) as f:
#            f.info()
#        print(item.name, (f[0].header['EXPTIME']))
            data = fits.getdata(basepath/item.name)
            SMeans.append(data.mean())
            SExpos.append(f[0].header['EXPTIME'])
#            print("mean: {}, exp: {}".format(data.mean(), (f[0].hea
#            plt.scatter((f[0].header['EXPTIME']),data.mean())

plt.scatter(SExpos,SMeans)
plt.title('Linearity')
plt.xlabel('Seconds')
plt.ylabel('Counts')
plt.ylim(0,70000)



# READ Med Files

#from pathlib import Path
MMeans = []
MExpos = []

basepath = Path('2020-01-23/Med/')
files_in_basepath = basepath.iterdir()
for item in files_in_basepath:
    if item.is_file():
        with fits.open(item) as f:
#            f.info()
#        print(item.name, (f[0].header['EXPTIME']))
            data = fits.getdata(basepath/item.name)
            MMeans.append(data.mean())
            MExpos.append(f[0].header['EXPTIME'])
#            print("mean: {}, exp: {}".format(data.mean(), (f[0].hea
#            plt.scatter((f[0].header['EXPTIME']),data.mean())

plt.scatter(MExpos,MMeans)
#plt.title('Linearity')
#plt.xlabel('Seconds')
```

```
#plt.ylabel('Counts')
#plt.ylim(0,70000)

# READ Fast Files

#from pathlib import Path
FMeans = []
FExpos = []

basepath = Path('2020-01-23/Fast/')
files_in_basepath = basepath.iterdir()
for item in files_in_basepath:
    if item.is_file():
        with fits.open(item) as f:
#             f.info()
#         print(item.name, (f[0].header['EXPTIME']))
            data = fits.getdata(basepath/item.name)
            FMeans.append(data.mean())
            FExpos.append(f[0].header['EXPTIME'])
#             print("mean: {}, exp: {}".format(data.mean(), (f[0].hea
#             plt.scatter((f[0].header['EXPTIME']),data.mean())

plt.scatter(FExpos,FMeans)
#plt.title('Linearity')
#plt.xlabel('Seconds')
#plt.ylabel('Counts')
#plt.ylim(0,70000)
plt.savefig('Lab1_DarkCurComp.png')
plt.show

```

Out[17]: <function matplotlib.pyplot.show(*args, **kw)>



In [ ]:
```
1
```