In [2]:
```
import datetime
import plotly.graph_objs as go
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplo
t
import pandas as pd
import json
import matplotlib.pyplot as plt

init_notebook_mode(connected=True)
```

Constants

In [3]:
```
INPUT_FILE = "/home/leenix/Active Projects/bridge-puck/test.log"
TIMESTAMP_FORMAT = "%Y-%m-%dT%H:%M:%S"

TIMESTAMP_CORRECTION_HOURS = 10
```

Grab data file

In [4]:
```
with open(INPUT_FILE) as f:
    lines = f.readlines()

entries = []
micros = 0
for l in lines:
    l = l.strip('\n')
    l = l.replace('\'', '\"')

    # Correct timestamps
    e = json.loads(l)
    if 'micros' in e.keys():
        micros = e["micros"]
    e['datetime'] = datetime.datetime.strptime(e['time'], TIMESTAMP_FORMAT)
+ datetime.timedelta(hours=TIMESTAMP_CORRECTION_HOURS)
    e['datetime'] += datetime.timedelta(microseconds=micros)
    if 'offset' in e.keys():
        e['datetime'] += datetime.timedelta(microseconds=(e['offset']*1000))

    entries.append(e)

df = pd.DataFrame(entries)
df = df.sort_values(by='datetime')
df = df.drop(df[df['datetime'] < '2019-01-01'].index)
```

In [5]:
```
df.describe()
```

Out[5]:

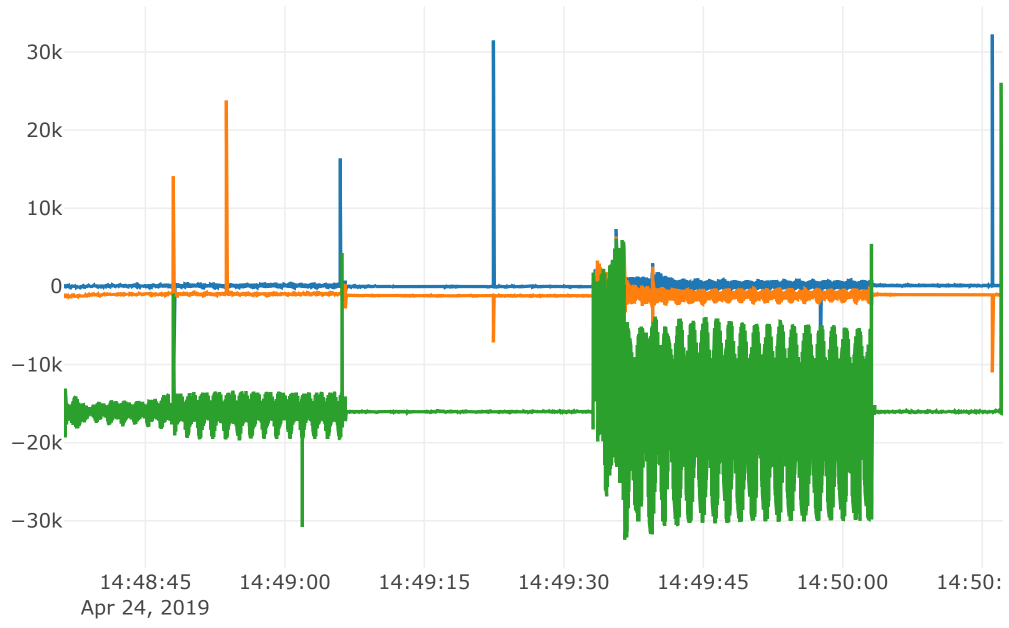
	micros	offset	temperature	v_battery	x	y	z
count	15.000000	13893.000000	11.000000	3.00000	13893.000000	13893.000000	13893.000000
mean	203568.133333	4958.714892	11.901989	-2.25400	134.349529	-1039.308501	-15409.278702
std	28.192873	2902.741312	45.472758	11.91651	606.946223	610.579941	4254.886718
min	203527.000000	0.000000	-125.203125	-16.01400	-16127.000000	-25663.000000	-32413.000000
25%	203552.500000	2417.000000	25.500000	-5.69400	-17.000000	-1181.000000	-16271.000000
50%	203563.000000	4945.000000	25.625000	4.62600	122.000000	-1045.000000	-16027.000000
75%	203577.500000	7476.000000	25.687500	4.62600	245.000000	-916.000000	-14607.000000
max	203633.000000	15647.000000	25.750000	4.62600	32251.000000	23804.000000	26049.000000

Quick plots

Acceleration

In [6]:
```
accel = df.loc[df['x'].notnull()].reset_index()[['datetime', 'x', 'y', 'z']]
```

```
In [7]: x = go.Scatter(x=accel['datetime'], y=accel.x, name="x")
        y = go.Scatter(x=accel.datetime, y=accel.y, name="y")
        z = go.Scatter(x=accel.datetime, y=accel.z, name="z")

        data = [x,y,z]

        layout = go.Layout()
        fig = go.Figure(data=data, layout=layout)

        iplot(fig)
```



Strain

```
In [8]: strain = df.loc[df['strain'].notnull()].reset_index().loc[:, ['datetime', 's
        train']]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
~/.local/share/virtualenvs/Analysis-rYV1M2Ne/lib/python3.6/site-packages/pand
as/core/indexes/base.py in get_loc(self, key, method, tolerance)
   2656             try:
-> 2657                 return self._engine.get_loc(key)
   2658             except KeyError:

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHas
hTable.get_item()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHas
hTable.get_item()

KeyError: 'strain'

During handling of the above exception, another exception occurred:

KeyError                                  Traceback (most recent call last)
<ipython-input-8-a556600a65d4> in <module>
----> 1 strain = df.loc[df['strain'].notnull()].reset_index().loc[:, ['dateti
me', 'strain']]

~/.local/share/virtualenvs/Analysis-rYV1M2Ne/lib/python3.6/site-packages/pand
as/core/frame.py in __getitem__(self, key)
   2925             if self.columns.nlevels > 1:
   2926                 return self._getitem_multilevel(key)
-> 2927             indexer = self.columns.get_loc(key)
   2928             if is_integer(indexer):
   2929                 indexer = [indexer]

~/.local/share/virtualenvs/Analysis-rYV1M2Ne/lib/python3.6/site-packages/pand
as/core/indexes/base.py in get_loc(self, key, method, tolerance)
   2657                 return self._engine.get_loc(key)
   2658             except KeyError:
-> 2659                 return self._engine.get_loc(self._maybe_cast_indexer(
key))
   2660         indexer = self.get_indexer([key], method=method, tolerance=to
lerance)
   2661         if indexer.ndim > 1 or indexer.size > 1:

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHas
hTable.get_item()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHas
hTable.get_item()

KeyError: 'strain'
```

In [9]:
```
initial_strain = strain.iloc[0]['strain']
strain.loc[:, 'strain_diff'] = initial_strain + strain['strain']
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-9-3f5f8b9281f0> in <module>
----> 1 initial_strain = strain.iloc[0]['strain']
      2 strain.loc[:, 'strain_diff'] = initial_strain + strain['strain']

NameError: name 'strain' is not defined
```

In [10]:
```
raw_strain = go.Scatter(x=strain['datetime'], y=strain['strain'], name="Stra
in")
diff_strain = go.Scatter(x=strain['datetime'], y=strain['strain_diff'], name
="Strain (diff)")
data = [raw_strain, diff_strain]

layout = go.Layout(xaxis=dict(title="Date"), yaxis=dict(title="Strain"))

fig = go.Figure(data=data, layout=layout)
iplot(fig)
```

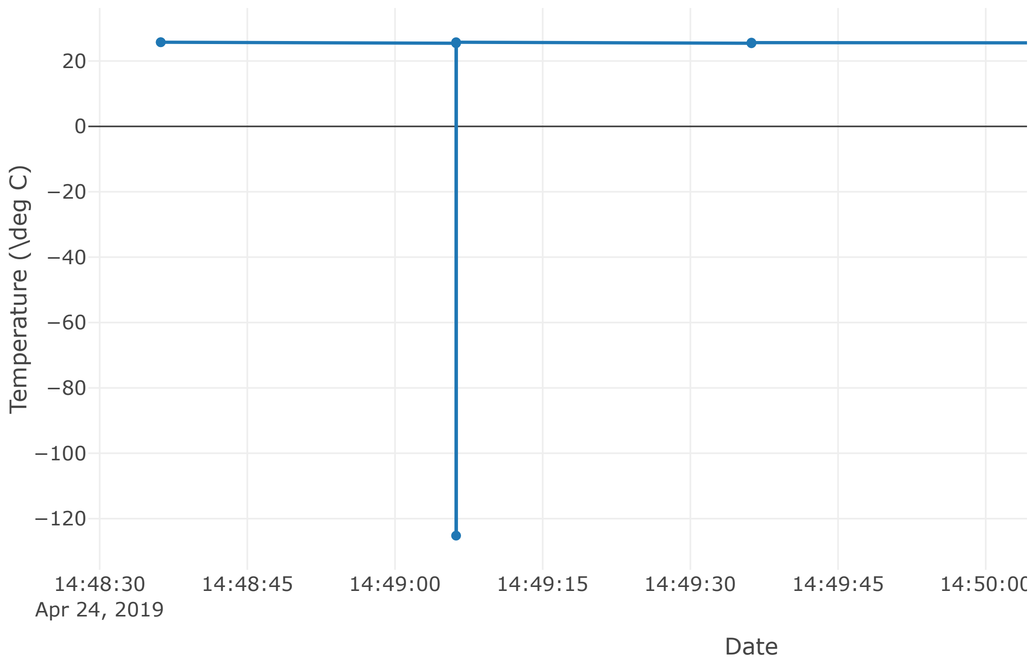
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-10-f144c149b933> in <module>
----> 1 raw_strain = go.Scatter(x=strain['datetime'], y=strain['strain'], nam
e="Strain")
      2 diff_strain = go.Scatter(x=strain['datetime'], y=strain['strain_diff'
], name="Strain (diff)")
      3 data = [raw_strain, diff_strain]
      4 
      5 layout = go.Layout(xaxis=dict(title="Date"), yaxis=dict(title="Strain
"))

NameError: name 'strain' is not defined
```

Temperature

In [11]:
```
temperature = df[df["temperature"].notnull()][df["temperature"] != 0][["temp
erature", "datetime"]].sort_values(by='datetime')
```

```
/home/leenix/.local/share/virtualenvs/Analysis-rYV1M2Ne/lib/python3.6/site-pa
ckages/ipykernel_launcher.py:1: UserWarning:

Boolean Series key will be reindexed to match DataFrame index.
```

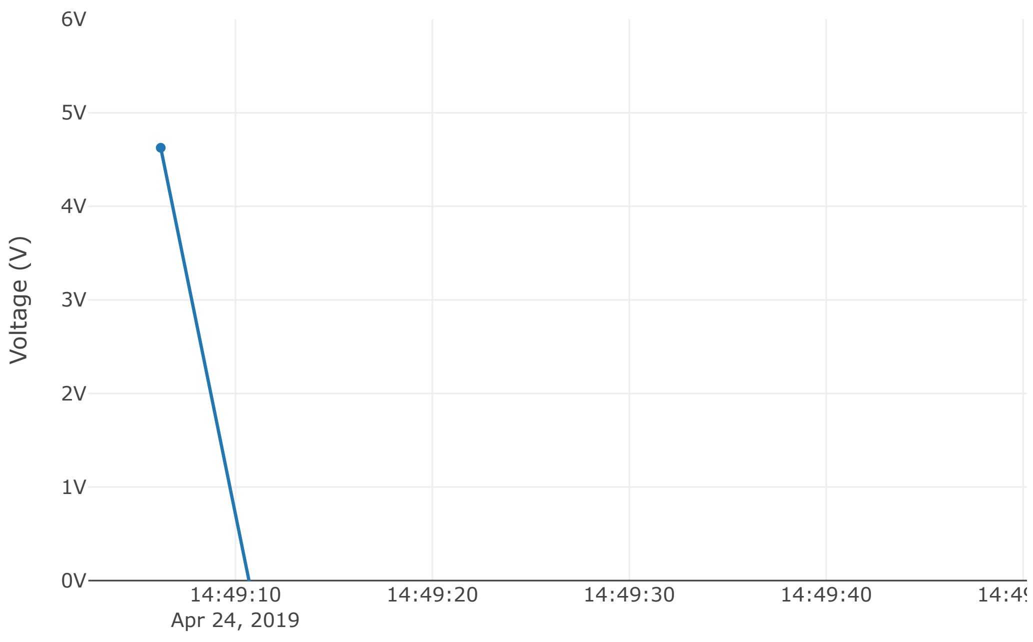
In [12]:
```
t = go.Scatter(x=temperature['datetime'], y=temperature['temperature'], name
="Case Temperature")

l = go.Layout(xaxis=dict(title="Date"), yaxis=dict(title="Temperature (\deg
C)"))

f = go.Figure(data=[t], layout=l)
iplot(f)
```



Battery

In [13]:
```
battery = df.loc[df['v_battery'].notnull(), ['datetime', 'v_battery']]
```

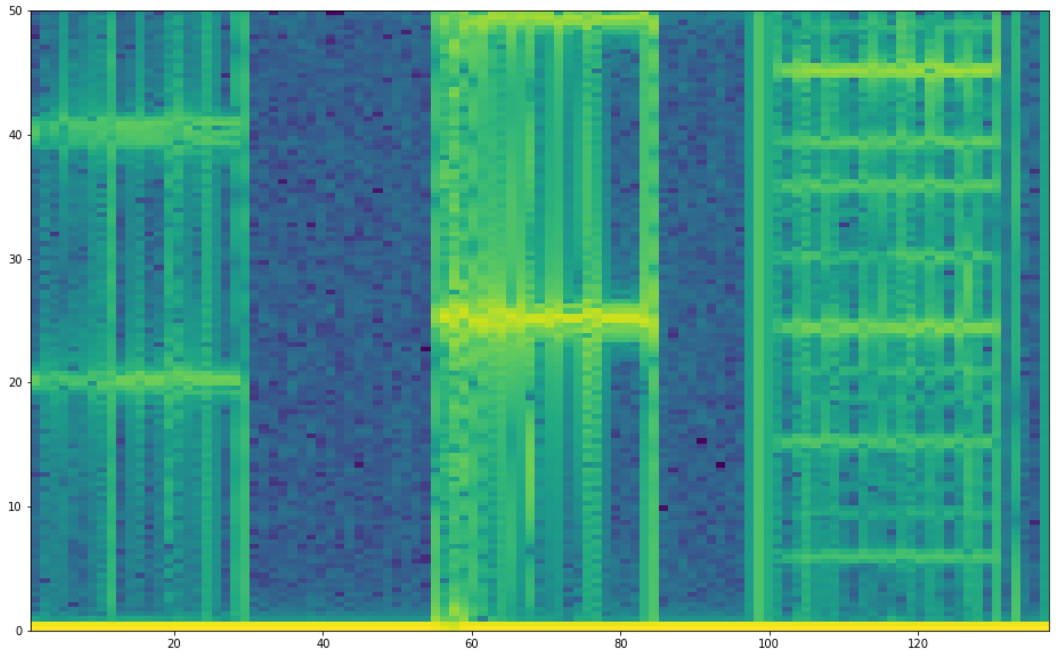
In [14]:
```
p = go.Scatter(x=battery['datetime'], y=battery['v_battery'], name="Battery
Voltage")
layout = go.Layout(yaxis=dict(title="Voltage (V)", ticksuffix="V", range=(0,
6)))
fig = go.Figure(data=[p], layout=layout)
iplot(fig)
```



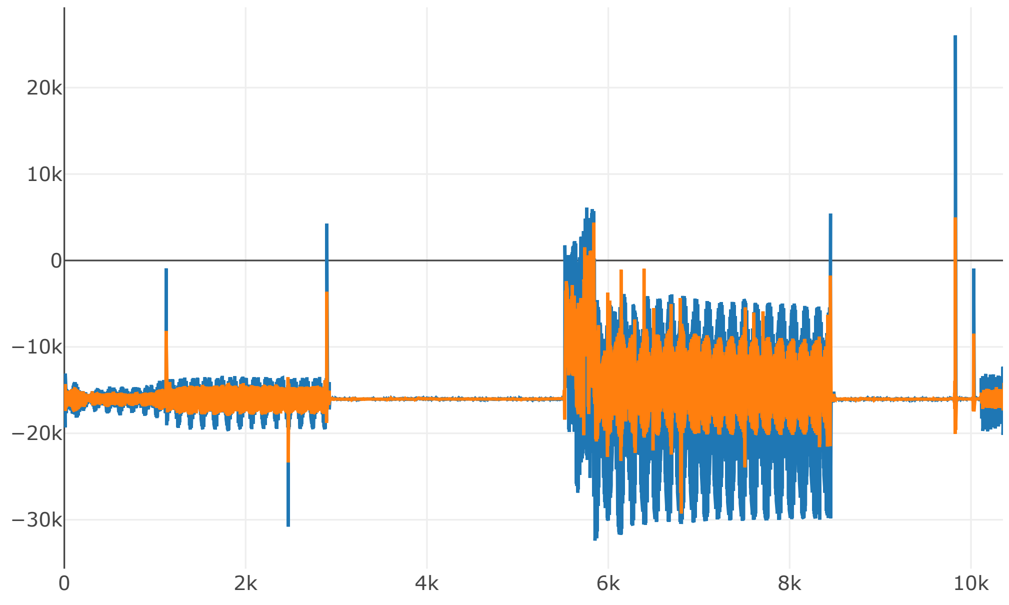
Plot spectrogram

In [15]:
```
import scipy.signal as signal
import matplotlib.pyplot as plt
import numpy as np
```

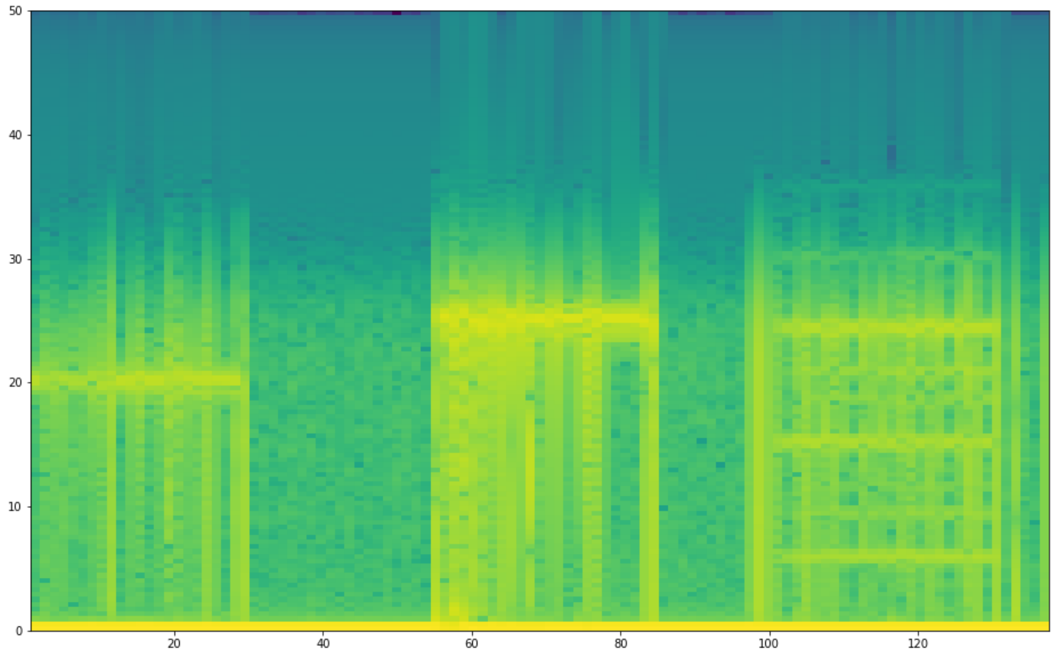
In [41]:
```
fig, ax = plt.subplots(figsize=(16,10))
ax.specgram(accel['z'], Fs=100, NFFT=256, noverlap=128)
plt.show()
```



In [46]:
```
def butter_lowpass(cutoff, fs, order=5):
    nyq = 0.5 * fs
    normal_cutoff = cutoff / nyq
    b, a = signal.butter(order, normal_cutoff, btype='low', analog=False)
    return b, a

def butter_lowpass_filter(data, cutoff, fs, order=5):
    b, a = butter_lowpass(cutoff, fs, order=order)
    y = lfilter(b, a, data)
    return y

# Filter requirements.
order = 6
fs = 100       # sample rate, Hz
cutoff = 25  # desired cutoff frequency of the filter, Hz

B, A = signal.butter(order, (2*cutoff)/fs, output='ba', analog=False)
smoothed = signal.filtfilt(B, A, accel['z'])

smooth = go.Scatter(y=smoothed, name="Smooth")
raw = go.Scatter(y=accel['z'], name="Raw")
data = [raw, smooth]
l = go.Layout()
fig = go.Figure(data=data, layout=l)
iplot(fig)
```



In [52]:
```
fig, ax = plt.subplots(figsize=(16,10))
ax.specgram(smoothed, Fs=100, NFFT=256, noverlap=128)
plt.show()
```


50
40
30
20
10
0
20
40
60
80
100
120