

```

In [163... %matplotlib notebook

import matplotlib.pyplot as plt #import matplotlib library
from matplotlib.animation import FuncAnimation # import animations for matplot

from datetime import datetime # datetime library

import pandas as pd # pandas library

from sqlalchemy.dialects.postgresql import JSON # Allows SQLAlchemy to parse t
from sqlalchemy import ( # Sqlalchemy desired content
    MetaData,
    Table, Column,
    Integer, Numeric, String,
    DateTime,
    ForeignKey,
    Select,
    create_engine)
from sqlalchemy.orm import sessionmaker # Sqlalchemy ORM desired content
from sqlalchemy.orm.exc import NoResultFound # Sqlalchemy NoResultFound

from time import sleep # sleep

In [164... metadata = MetaData() # Set SQLAlchemy MetaData

In [165... # Obtain database password
dbpass = ''
with open('../lunacapture/.dbpass') as f:
    dbpass = f.readlines()

In [166... # Create the SQLAlchemy Engine
engine = create_engine('postgresql://postgres:' + dbpass[0] + '@localhost:5432

In [167... # Create the connection to the engine
connection = engine.connect()

In [168... # Create a SQLAlchemy Session
Session = sessionmaker(bind = engine)
session = Session()

In [169... # Build a SQLAlchemy model of the PostgreSQL database
test_conn = Table('test_conn', metadata,
                  Column('id', Integer(), primary_key=True),
                  Column('instance_num', Integer()),
                  Column('robot_json', JSON),
                  Column('created_at', DateTime(timezone=False), default=datet

In [170... # List of possible data points to observe
data_record_names = [
    'boom',
    'drive_encoder_left',
    'drive_encoder_right',
    'dump',
    'epoch_time',
    'fork',

```

```

'loc_angle',
'loc_x',
'loc_y',
'power_boom',
'power_dump',
'power_fork',
'power_left',
'power_right',
'power_spin',
'power_stick',
'power_tilt',
'power_tool',
'spin',
'state_state',
'stick',
'tilt',
'vibe'
]

```

```

In [171...] sub_plot_0_name = data_record_names[10]
            sub_plot_1_name = data_record_names[3]
            sub_plot_2_name = data_record_names[19]

```

```

In [172...] print("Sub Plot 1: ", sub_plot_0_name, "\nSub Plot 2: ", sub_plot_1_name, "\nS
            Sub Plot 1: power_dump
            Sub Plot 2: dump
            Sub Plot 3: state_state

```

```

In [173...] # Create an empty pandas dataframe with columns to match
            df = pd.DataFrame()

            df['id'] = ''
            df['instance_num'] = ''
            df[sub_plot_0_name] = ''
            df[sub_plot_1_name] = ''
            df[sub_plot_2_name] = ''
            df['datetime_database'] = ''

```

```

In [174...] def print_data(show_print, results):
            if (not show_print):
                return
            else:
                for result in results:
                    print(result[0], ", ", result[1], ", ", result[2][sub_plot_0_name])
                return

```

```

In [175...] def obtain_instance_num():
            results = session.query(test_conn).order_by(test_conn.c.id.desc()).first()
            return results[1]

```

```

In [176...] # To have the most recent instance_num variable for this session, use the foll
            curr_instance_num = obtain_instance_num()
            print(curr_instance_num)

            # To set an instance_num variable manually,
            # curr_instance_num = 1

```

3

```
In [177... def retrieve_data(show_print):  
    # Prepare to select from the test_conn table  
    results = None  
  
    if df.empty:  
        s = test_conn.select().where(test_conn.c.instance_num == curr_instance  
        rp = connection.execute(s)  
        results = rp.fetchall()  
        if (results == None):  
            raise SystemExit("The postgresql database is empty.")  
        print_data(show_print, results)  
    else:  
        results = session.query(test_conn).filter(test_conn.c.instance_num ==  
        if (results == None):  
            raise SystemExit("The postgresql database is empty.")  
        print_data(show_print, [results])  
  
    # If the dataframe is empty, add this first result  
    if (len(df) == 0):  
        # To Do: Lift below into its own function, so as not to repeat  
        for result in results:  
            df.loc[len(df)] = [result[0], result[1], result[2][sub_plot_0_name  
  
    # To Do: Change so that all results that are not yet in dataframe are adde  
    # up to the last one added  
  
    # Otherwise, only add if the id in the result is different from the last i  
    elif (results[0] != df.iloc[-1]['id']):  
        df.loc[len(df)] = [results[0], results[1], results[2][sub_plot_0_name]
```

```
In [178... retrieve_data(True)
```

24	,	3	,	0.0	,	6.489999771118164	,	2	,	2023-05-20	22:55:58.543312
25	,	3	,	0.0	,	6.489999771118164	,	2	,	2023-05-20	22:55:59.047843
26	,	3	,	0.0	,	6.489999771118164	,	2	,	2023-05-20	22:55:59.550969
27	,	3	,	0.0	,	6.489999771118164	,	2	,	2023-05-20	22:56:00.056368
28	,	3	,	0.0	,	6.489999771118164	,	2	,	2023-05-20	22:56:00.562299
29	,	3	,	0.0	,	6.489999771118164	,	2	,	2023-05-20	22:56:01.067767
30	,	3	,	0.0	,	0.0	,	2	,	2023-05-20	22:56:01.572209
31	,	3	,	0.0	,	0.0	,	2	,	2023-05-20	22:56:02.076517
32	,	3	,	0.0	,	0.0	,	2	,	2023-05-20	22:56:02.581312
33	,	3	,	0.0	,	0.0	,	2	,	2023-05-20	22:56:03.085858
34	,	3	,	0.0	,	0.0	,	2	,	2023-05-20	22:56:03.591326
35	,	3	,	-0.0	,	-1.2400000095367432	,	2	,	2023-05-20	22:56:04.096734
36	,	3	,	-0.0	,	-1.25	,	2	,	2023-05-20	22:56:04.600530
37	,	3	,	-0.0	,	-1.25	,	2	,	2023-05-20	22:56:05.104523
38	,	3	,	-0.0	,	-1.25	,	2	,	2023-05-20	22:56:05.609945
39	,	3	,	-0.0	,	-1.25	,	2	,	2023-05-20	22:56:06.115856
40	,	3	,	-0.0	,	-1.25	,	2	,	2023-05-20	22:56:06.620621
41	,	3	,	-0.0	,	-1.25	,	2	,	2023-05-20	22:56:07.124936
42	,	3	,	-0.0	,	-1.25	,	2	,	2023-05-20	22:56:07.630411
43	,	3	,	-0.0	,	-1.25	,	2	,	2023-05-20	22:56:08.135817
44	,	3	,	-0.0	,	-1.25	,	2	,	2023-05-20	22:56:08.640637
45	,	3	,	-0.0	,	-1.25	,	2	,	2023-05-20	22:56:09.144928
46	,	3	,	-0.0	,	-1.25	,	2	,	2023-05-20	22:56:09.652206
47	,	3	,	-0.0	,	-1.25	,	2	,	2023-05-20	22:56:10.157607
48	,	3	,	-0.0	,	-1.25	,	2	,	2023-05-20	22:56:10.662589
49	,	3	,	-0.0	,	-1.25	,	2	,	2023-05-20	22:56:11.167286
50	,	3	,	-0.0	,	-1.25	,	2	,	2023-05-20	22:56:11.673322
51	,	3	,	-0.0	,	-1.25	,	2	,	2023-05-20	22:56:12.178295
52	,	3	,	-0.0	,	-1.25	,	2	,	2023-05-20	22:56:12.682690
53	,	3	,	-0.0	,	-1.25	,	2	,	2023-05-20	22:56:13.188198
54	,	3	,	-0.0	,	-1.25	,	2	,	2023-05-20	22:56:13.693925
55	,	3	,	-0.0	,	-1.25	,	2	,	2023-05-20	22:56:14.197606
56	,	3	,	-0.0	,	-1.25	,	2	,	2023-05-20	22:56:14.701884
57	,	3	,	-0.0	,	-1.25	,	2	,	2023-05-20	22:56:15.207755
58	,	3	,	-0.0	,	-1.25	,	2	,	2023-05-20	22:56:15.712686
59	,	3	,	-0.0	,	-1.25	,	2	,	2023-05-20	22:56:16.217598
60	,	3	,	-0.0	,	-1.25	,	2	,	2023-05-20	22:56:16.721681
61	,	3	,	-0.0	,	-1.25	,	2	,	2023-05-20	22:56:17.227067
62	,	3	,	-0.0	,	-1.25	,	2	,	2023-05-20	22:56:17.732476
63	,	3	,	-0.0	,	-1.25	,	2	,	2023-05-20	22:56:18.235957
64	,	3	,	-0.0	,	-1.25	,	2	,	2023-05-20	22:56:18.740639
65	,	3	,	-0.0	,	-1.25	,	2	,	2023-05-20	22:56:19.246151
66	,	3	,	-0.0	,	-1.25	,	2	,	2023-05-20	22:56:19.751006
67	,	3	,	-0.0	,	-1.25	,	2	,	2023-05-20	22:56:20.255551
68	,	3	,	-0.0	,	-1.25	,	2	,	2023-05-20	22:56:20.760827
69	,	3	,	-0.0	,	-1.25	,	2	,	2023-05-20	22:56:21.266448
70	,	3	,	-0.0	,	-1.25	,	2	,	2023-05-20	22:56:21.771850
71	,	3	,	-0.0	,	-1.25	,	2	,	2023-05-20	22:56:22.276177
72	,	3	,	-0.0	,	-1.25	,	2	,	2023-05-20	22:56:22.781520
73	,	3	,	-0.0	,	-1.25	,	2	,	2023-05-20	22:56:23.286266
74	,	3	,	-0.0	,	-1.25	,	2	,	2023-05-20	22:56:23.790561
75	,	3	,	-0.0	,	-1.25	,	2	,	2023-05-20	22:56:24.295214
76	,	3	,	-0.0	,	-1.25	,	2	,	2023-05-20	22:56:24.800987
77	,	3	,	-0.0	,	-1.25	,	2	,	2023-05-20	22:56:25.306503
78	,	3	,	-0.0	,	-1.25	,	2	,	2023-05-20	22:56:25.811836
79	,	3	,	-0.0	,	-1.25	,	2	,	2023-05-20	22:56:26.317647
80	,	3	,	-0.0	,	-1.25	,	2	,	2023-05-20	22:56:26.823656
81	,	3	,	-0.0	,	-1.25	,	2	,	2023-05-20	22:56:27.329082
82	,	3	,	-0.0	,	-1.25	,	2	,	2023-05-20	22:56:27.835097
83	,	3	,	-0.0	,	-1.25	,	2	,	2023-05-20	22:56:28.340374

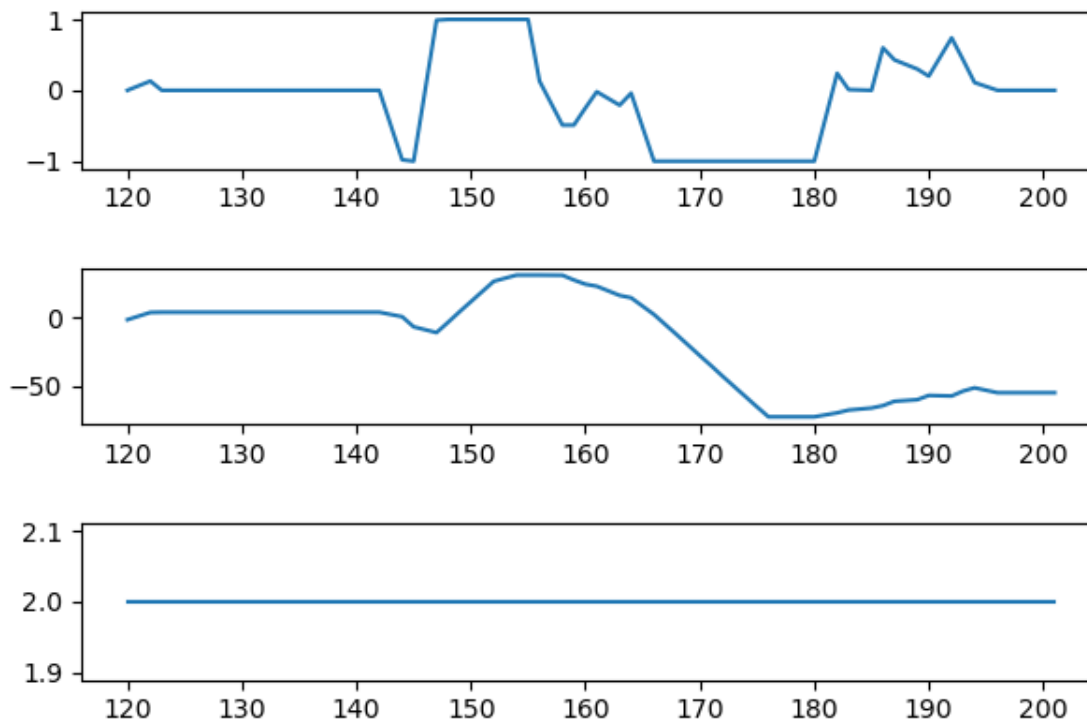
```
84 , 3 , -0.0 , -1.25 , 2 , 2023-05-20 22:56:28.844502
85 , 3 , -0.0 , -1.25 , 2 , 2023-05-20 22:56:29.349080
86 , 3 , -0.0 , -1.25 , 2 , 2023-05-20 22:56:29.852643
87 , 3 , -0.0 , -1.25 , 2 , 2023-05-20 22:56:30.358203
88 , 3 , -0.0 , -1.25 , 2 , 2023-05-20 22:56:30.863387
89 , 3 , -0.0 , -1.25 , 2 , 2023-05-20 22:56:31.368714
90 , 3 , -0.0 , -1.25 , 2 , 2023-05-20 22:56:31.874121
91 , 3 , -0.0 , -1.25 , 2 , 2023-05-20 22:56:32.379418
```

```
In [179... def update_graph_length(graph_initial_cell, graph_x_length):
            if (len(df) > graph_x_length):
                graph_initial_cell = len(df) - graph_x_length
            return graph_initial_cell
```

```
In [180... fig, axs = plt.subplots(3)
fig.suptitle("Excavator Data Feed")
plt.ion()

fig.show()
fig.canvas.draw()
```

Excavator Data Feed



```
In [181... # Initialize the graph starting point at 0
graph_initial_cell = 0

# Set the intended graph length
graph_x_length = 60
```

```
In [182... print(df)
```

	id	instance_num	power_dump	dump	state_state	datetime_database
0	24	3	0.0	6.49	2	2023-05-20 22:55:58.543312
1	25	3	0.0	6.49	2	2023-05-20 22:55:59.047843
2	26	3	0.0	6.49	2	2023-05-20 22:55:59.550969
3	27	3	0.0	6.49	2	2023-05-20 22:56:00.056368
4	28	3	0.0	6.49	2	2023-05-20 22:56:00.562299
..
63	87	3	-0.0	-1.25	2	2023-05-20 22:56:30.358203
64	88	3	-0.0	-1.25	2	2023-05-20 22:56:30.863387
65	89	3	-0.0	-1.25	2	2023-05-20 22:56:31.368714
66	90	3	-0.0	-1.25	2	2023-05-20 22:56:31.874121
67	91	3	-0.0	-1.25	2	2023-05-20 22:56:32.379418

[68 rows x 6 columns]

```
In [183... # Endless while loop to display graph
while True:
    retrieve_data(False)

    axs[0].clear()
    axs[1].clear()
    axs[2].clear()

    graph_initial_cell = update_graph_length(graph_initial_cell, graph_x_lengt

    axs[0].plot(df.iloc[graph_initial_cell:len(df)]['id'], df.iloc[graph_initi
    axs[1].plot(df.iloc[graph_initial_cell:len(df)]['id'], df.iloc[graph_initi
    axs[2].plot(df.iloc[graph_initial_cell:len(df)]['id'], df.iloc[graph_initi

#     axs[0].set_title(sub_plot_0_name, rotation=90, x=-0.2, y=0.1)
#     axs[1].set_title(sub_plot_1_name, rotation=90, x=-0.2, y=0.1)
#     axs[2].set_title(sub_plot_2_name, rotation=90, x=-0.2, y=0.1)

fig.tight_layout(pad=2.0)

fig.canvas.draw()

sleep(500/1000)
# clear_output(wait=True)
```

```
-----
KeyboardInterrupt                               Traceback (most recent call last)
/tmp/ipykernel_43772/2410811913.py in <module>
     22     fig.canvas.draw()
     23
--> 24     sleep(500/1000)
     25     # clear_output(wait=True)

KeyboardInterrupt:
```

In []: