In [ ]:
```
'''Jonathan Zhang'''
```

Checkpoint #1 Report:

Could not figure out how to make run.py run all my tests for Battery and Device Use.

Thought to compile an encompassing notebook to show you the data analysis we performed. Thank you!

Problem to investigate:

- "What parameters can help with battery health related issues?"

Based on the data/samples collected from the Battery DLL file, we found some parameters that we believe can influence battery health substantially.

- ACPI-BATTERY(1) - # of batteries detected
- ACPI-BATTERY(2) - Battery life %
- ACPI-BATTERY(6) - Chemistry
- ACPI-BATTERY(7) - Estimated runtime (s)
- ACPI-BATTERY(8) - Charge discharge rate
- ACPI-BATTERY(10) - Full charge capacity

In our subject domain, we utilize Intel's SDK (software development kit) for data collection. In doing so, we use their input libraries to extrapolate data from our computer's user wait time, processes, device and battery usage, specifically BATTERY and DEVICE USE.

In order to generate the data we needed to accomplish the tasks at hand, specifically performing data analysis on Battery and Device Use, we needed to use Intel's ESRV process which essentially gave us all the data we needed from our own computers. The output database file contained the timestamp at which the battery life was at a certain percentage, along with other various statistics and characteristics of our battery. We believe that in analyzing the parameters above, we can substantiate findings to see whether or not we can prolong battery life.

Checkpoint #1 Code for BATTERY:

Imports

In [2]:
```
import numpy as np
import pandas as pd
```

Data Preparation

In [3]:
```
# Data for BATTERY
data = pd.read_csv("../data/temp/counters_ull_time_data.csv")
data_2 = pd.read_csv("../data/temp/counters_double.csv")

# Data for DEVICE USE
du_data = pd.read_csv("../data/temp/du_counters_ull_time_data.csv")
du_data_2 = pd.read_csv("../data/temp/du_counters_string_time_data.csv")
```

COUNTERS_ULL_TIME_DATA.csv -- Data Analysis

In [4]:
```
# Data Demographic
    # 0 : ACDC
    # 1 : BATTERY COUNT
    # 2 : BATTERY LIFE
    # 3 : CAPABILITIES
    # 6 : ESTIMATED RUN TIME
    # 8 : THEORETICAL CAPACITY
    # 9 : FULL_CHARGE_CAPACITY
data['ID_INPUT'].value_counts()
```

Out[4]:
```
6    1403
2      38
9       1
8       1
3       1
1       1
0       1
Name: ID_INPUT, dtype: int64
```

We think that the battery life(%) is an important parameter for we to investigate our question. Since the ID input of battery_life is 2, we will let the ID_INPUT equal to 2 and analyze the output.

In [5]:
```
# All the "VALUE(s)" presented when ID_INPUT is 2.
battery_life = data.loc[(data['ID_INPUT'] == 2)]
```

In [6]:
```
battery_life.head()
```

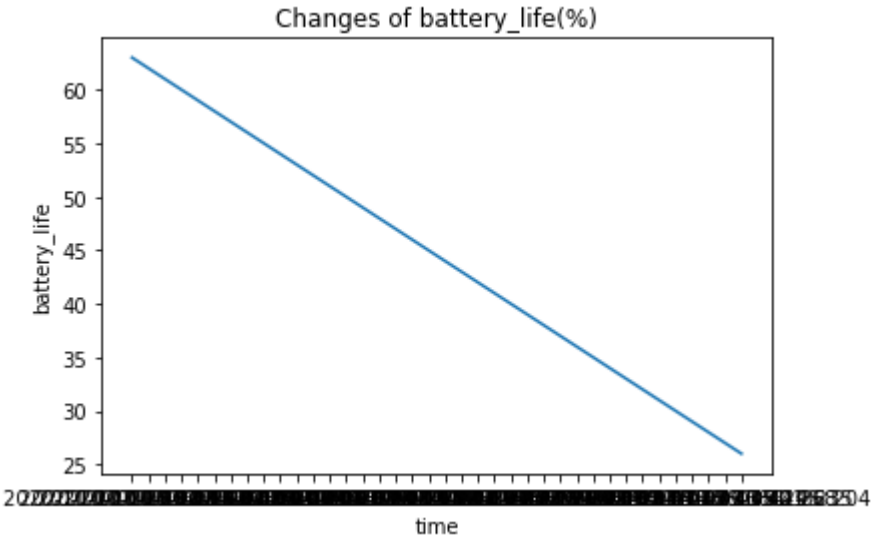
Out[6]:

	MEASUREMENT_TIME	ID_INPUT	VALUE	PRIVATE_DATA
2	2020-11-07 00:51:22.229	2	63	0
34	2020-11-07 00:52:17.208	2	62	0
72	2020-11-07 00:53:43.188	2	61	0
111	2020-11-07 00:55:07.162	2	60	0
142	2020-11-07 00:56:17.142	2	59	0

For more clear to see the change of battery_life value, we would use line chart and set the x-axis as time and y-axis as the value of battery_life

In [7]:
```
import matplotlib.pyplot as plt
```

In [8]:
```
x = battery_life['MEASUREMENT_TIME']
y = battery_life['VALUE']
plt.plot(x,y)
plt.title('Changes of battery_life(%)')
plt.xlabel('time')
plt.ylabel('battery_life')
plt.show()
```



In [9]:
```
# Max value when ID_INPUT is 2
data.loc[(data['ID_INPUT'] == 2)].max()
```

Out[9]:
```
MEASUREMENT_TIME    2020-11-07 01:44:36.204
ID_INPUT                                  2
VALUE                                    63
PRIVATE_DATA                              0
dtype: object
```

In [10]:
```
# Min value when ID_INPUT is 2
data.loc[(data['ID_INPUT'] == 2)].min()
```

Out[10]:
```
MEASUREMENT_TIME    2020-11-07 00:51:22.229
ID_INPUT                                  2
VALUE                                    26
PRIVATE_DATA                              0
dtype: object
```

In [11]:
```
# Average Battery Life % is 44.5%
data.loc[(data['ID_INPUT'] == 2)].mean()
```

Out[11]:
```
ID_INPUT         2.0
VALUE           44.5
PRIVATE_DATA     0.0
dtype: float64
```

Conclusion: As we can see from above dataset and line chart, the battery_life keeps decreasing for the whole collecting process. From the beginning, the battery_life is 63

decreasing for the whole collecting process. From the beginning, the battery_life is 30 which is the max value and then in the end, the battery_life decreased to 26 which is the min value.

COUNTERS_DOUBLE.csv -- Data Analysis

Now we are using counters_double dataset which contains charge_discharge_rate and system_power_in_dc. First we would analysis the charge_discharge_rate to see the changes of this parameter.

In [12]:
```
# Data Demographic
    # 7 : CHARGE_DISCHARGE_RATE
    # 10 : SYSTEM_POWER_IN_DC:WATTS:
data_2['ID_INPUT'].value_counts()
```

Out[12]:
```
7     1388
10     541
Name: ID_INPUT, dtype: int64
```

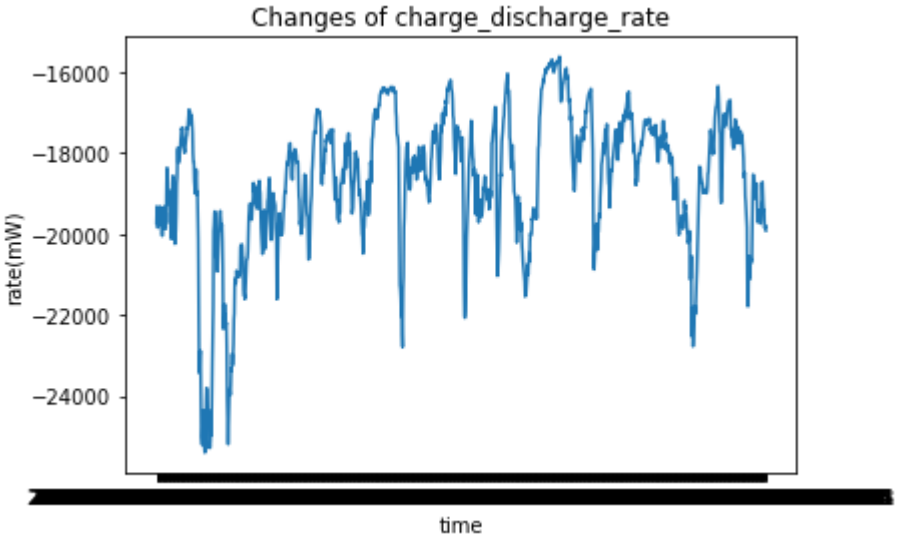
In [13]:
```
# All the "VALUE(s)" presented when ID_INPUT is 7.
charge_discharge = data_2.loc[(data_2['ID_INPUT'] == 7)]
charge_discharge.head()
```

Out[13]:

	MEASUREMENT_TIME	ID_INPUT	VALUE	PRIVATE_DATA
0	2020-11-07 00:51:22.229	7	-19725.0	1
1	2020-11-07 00:51:23.228	7	-19319.0	1
2	2020-11-07 00:51:25.226	7	-19813.0	1
3	2020-11-07 00:51:26.225	7	-19846.0	1
4	2020-11-07 00:51:27.225	7	-19807.0	1

We would also use the line chart to see the change of values

In [14]:
```
x = charge_discharge['MEASUREMENT_TIME']
y = charge_discharge['VALUE']
plt.plot(x,y)
plt.title('Changes of charge_discharge_rate')
plt.xlabel('time')
plt.ylabel('rate(mW)')
plt.show()
```



Conslusion: We think that since we didn't charge the laptop when collecting the data, the charge_discharge_rate would be negative. Also since we opened and closed several applications during the measurement time, the value of charge_discharge_rate changes inrregularly.

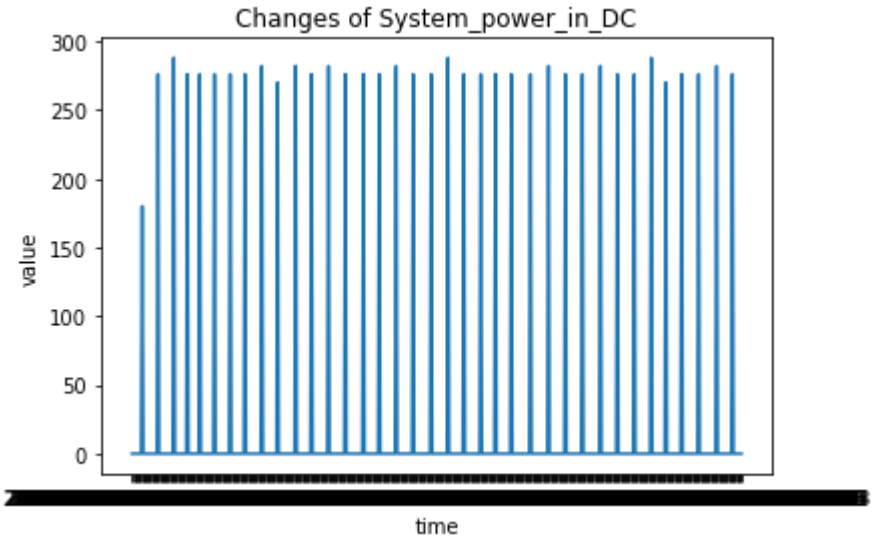
Then, we use the system_power_in_dc to see what are the values for this part if we didn't charge our laptop when collecting the data

In [15]:
```
# All the "VALUE(s)" presented when ID_INPUT is 10.
system_power = data_2.loc[(data_2['ID_INPUT'] == 10)]
system_power.head()
```

Out[15]:

	MEASUREMENT_TIME	ID_INPUT	VALUE	PRIVATE_DATA
5	2020-11-07 00:51:28.225	10	0.0	0
9	2020-11-07 00:51:34.223	10	0.0	0
13	2020-11-07 00:51:41.221	10	0.0	0
16	2020-11-07 00:51:46.217	10	0.0	0
21	2020-11-07 00:51:53.217	10	0.0	0

In [16]:
```
x = system_power['MEASUREMENT_TIME']
y = system_power['VALUE']
plt.plot(x,y)
plt.title('Changes of System_power_in_DC')
plt.xlabel('time')
plt.ylabel('value')
plt.show()
```



Conclusion: As we can see from the above chart, the system_power_in_dc has changed between 0 and over a hundred ...

In [17]:
```
# Max value when ID_INPUT is 7
    # Max value : -15621
data_2.loc[(data_2['ID_INPUT'] == 7)].max()
```

Out[17]:
```
MEASUREMENT_TIME    2020-11-07 01:45:30.191
ID_INPUT                                  7
VALUE                                 -15621
PRIVATE_DATA                               1
dtype: object
```

In [18]:
```
# Min value when ID_INPUT is 7
    # Min value : -25405
data_2.loc[(data_2['ID_INPUT'] == 7)].min()
```

Out[18]:
```
MEASUREMENT_TIME    2020-11-07 00:51:22.229
ID_INPUT                                  7
VALUE                                 -25405
PRIVATE_DATA                               1
dtype: object
```

In [19]:
```
# Max value when ID_INPUT is 10
    # Max value : 288.185
data_2.loc[(data_2['ID_INPUT'] == 10)].max()
```

Out[19]:
```
MEASUREMENT_TIME    2020-11-07 01:45:28.193
ID_INPUT                                 10
VALUE                                288.185
PRIVATE_DATA                               0
dtype: object
```

In [20]:
```
# Min value when ID_INPUT is 10
    # Min value : 0
data_2.loc[(data_2['ID_INPUT'] == 10)].min()
```

Out[20]:
```
MEASUREMENT_TIME    2020-11-07 00:51:28.225
ID_INPUT                                 10
VALUE                                     0
PRIVATE_DATA                              0
dtype: object
```

Checkpoint #1 Code for DEVICE USE:

In [21]:
```
# Real Numbers for VALUES
du_data.head(10)
```

Out[21]:

	MEASUREMENT_TIME	ID_INPUT	VALUE	PRIVATE_DATA
0	2020-11-07 01:50:13.775	3	4	1
1	2020-11-07 01:50:13.775	3	4	2
2	2020-11-07 01:50:13.775	3	4	3
3	2020-11-07 01:50:13.775	3	4	4
4	2020-11-07 01:50:13.775	3	4	5
5	2020-11-07 01:50:13.775	3	4	6
6	2020-11-07 01:50:13.775	3	4	7
7	2020-11-07 01:50:13.775	3	4	8
8	2020-11-07 01:50:13.775	3	4	9
9	2020-11-07 01:50:13.775	3	4	10

In [22]:
```
# Strings for VALUES
du_data_2.head(10)
```

Out[22]:

	MEASUREMENT_TIME	ID_INPUT	VALUE	PRIVATE_
0	2020-11-07 01:50:13.775	0	GUID_DEVINTERFACE_DISPLAY_ADAPTER	
1	2020-11-07 01:50:13.775	1	ROOT\BasicDisplay	
2	2020-11-07 01:50:13.775	2	Microsoft Basic Display Driver	
3	2020-11-07 01:50:13.775	0	GUID_DEVINTERFACE_DISPLAY_ADAPTER	
4	2020-11-07 01:50:13.775	1	PCI\VEN_8086&DEV_8A52&SUBSYS_385217AA&REV_07	
5	2020-11-07 01:50:13.775	2	Intel(R) Iris(R) Plus Graphics	
6	2020-11-07 01:50:13.775	0	GUID_DEVINTERFACE_MONITOR	
7	2020-11-07 01:50:13.775	1	MONITOR\BOE06F2	
8	2020-11-07 01:50:13.775	2	Generic PnP Monitor	
9	2020-11-07 01:50:13.775	0	GUID_DISPLAY_DEVICE_ARRIVAL	

In [23]:
```
# Device Use Data Demographic
    # 3 : Device Status
du_data["ID_INPUT"].value_counts()
```

Out[23]:
```
3    39
Name: ID_INPUT, dtype: int64
```

In [24]:
```
# Device Use Data Demographic
    # 0 : Device Guide
    # 1 : Device HW Name
    # 2 : Device Friendly Name
du_data_2['ID_INPUT'].value_counts()
```

Out[24]:
```
2    39
1    39
0    39
Name: ID_INPUT, dtype: int64
```

In [ ]:
```
"""----------------------------------------------------------------------------
```

Contributions:

Keshan:

- data preparation
- tabled data
- key notes all throughout notebook
- graphs + graph analysis

Jon:

- Report + main ideas
- data analysis - code breakdown
- repository structuring
- notebook outlining