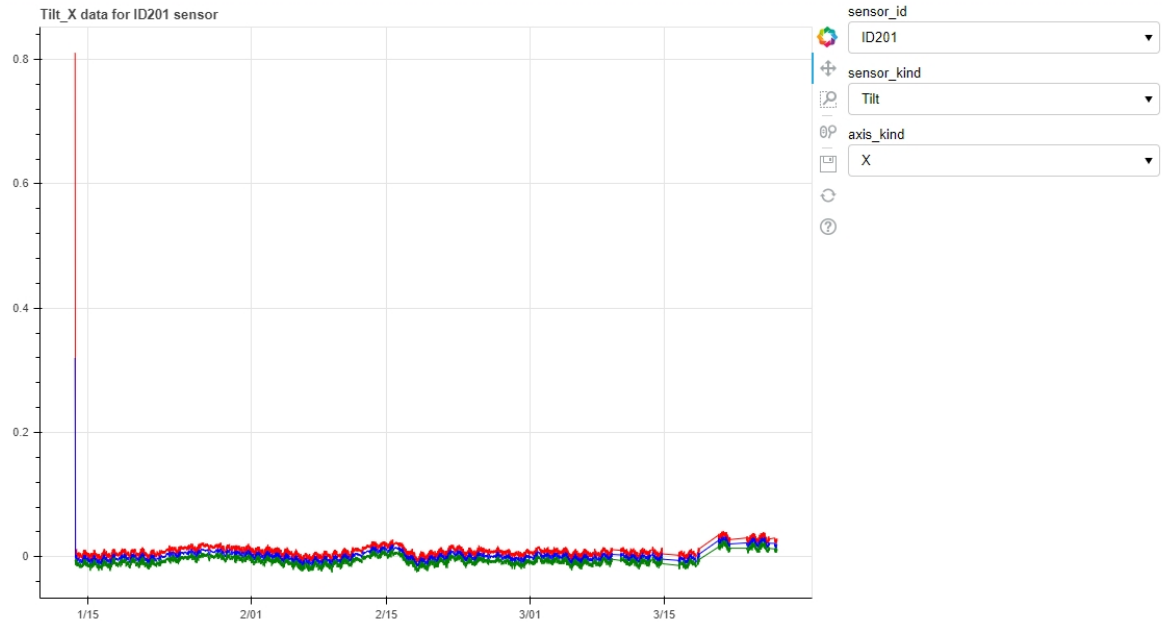
bokeh를 활용한 데이터 시각화

예시 사진



파일 실행

```
# 실행 시 터미널에서
$ bokeh serve 파일명.py
```

데이터 설명

모니터링 데이터에 4개의 센서모듈이 있으며, 각각의 모듈에는 2가지 종류(Tilt, Acel)의 센서가 내장되어 있음. 또한, Tilt 센서에는 X, Y 축 데이터가 존재하며 Acel 센서에는 X, Y, Z의 데이터가 존재한다.

- ID201 => GHID : 5
- ID202 => GHID : 6
- ID203 => GHID : 7
- ID204 => GHID : 8

기능

센서 모듈별, 센서 종류별, 축별 선택이 가능하도록 시각화 작업을 진행함. 옵션을 선택했을 때 선택된 `selected_value`를 넘겨 `key`값으로 접근하도록 구현.

```
# dropdown을 정의하기 위해 dict로 선택사항을 분류해서 지정해줌.
sensor_id_dict = {
    'ID201': {
        'GHID': 5,
        'title': 'ID201 sensor',
```

```
    },
    'ID202': {
        'GHID': 6,
        'title': 'ID202 sensor',
    },
    'ID203': {
        'GHID': 7,
        'title': 'ID203 sensor',
    },
    'ID204': {
        'GHID': 8,
        'title': 'ID204 sensor',
    },
}

sensor_kind_dict = {
    'Tilt': {
        'X': ['MXSineMax', 'MXSineAvg', 'MXSineMin'],
        'Y': ['MYSineMax', 'MYSineAvg', 'MYSineMin'],
    },
    'Acel': {
        'X': ['MXAcelMax', 'MXAcelAvg', 'MXAcelMin'],
        'Y': ['MYAcelMax', 'MYAcelAvg', 'MYAcelMin'],
        'Z': ['MZAcelMax', 'MZAcelAvg', 'MZAcelMin'],
    },
}

# dropdown 버튼을 생성함.
# 위에서 정의한 dict를 options으로 넘겨 줌.
sensor_id_select = Select(value=sensor_id, title='sensor_id',
options=sorted(sensor_id_dict.keys()))
sensor_kind_select = Select(value=sensor_kind, title='sensor_kind',
options=sorted(sensor_kind_dict.keys()))
axis_kind_select = Select(value=axis_kind, title='axis_kind', options=['X',
'Y'])
```

버튼을 선택했을 때 동적으로 변화를 주기위해, `update_plot` 이라는 함수를 정의해 사용.

```
# 동적으로 선택 옵션 적용
sensor_id_select.on_change('value', update_plot)
sensor_kind_select.on_change('value', update_plot)
axis_kind_select.on_change('value', update_plot)

# update_plot 함수
def update_plot(attrname, old, new):
    global layout

    sensor_id = sensor_id_select.value
    sensor_kind = sensor_kind_select.value
    axis_kind = axis_kind_select.value

    title = '{}_{}'.format(sensor_kind, axis_kind) + ' data for ' +
sensor_id_dict[sensor_id]['title']

    # 센서의 종류에 따라 축의 개수가 다르므로 재정의
```

```
    if sensor_kind_select.value == 'Acel':
        axis_kind_select.options = ['X', 'Y', 'Z']
    else:
        axis_kind_select.options = ['X', 'Y']

    # 선택한 옵션에 따라 데이터 셋을 재정의하고,
    # 새로 plotting 시켜 업데이팅한다.
    src = get_dataset(df, sensor_id_dict[sensor_id]['GHID'], sensor_kind,
axis_kind)
    layout.children[0] = make_plot(src, title)
```

LSTM

소스코드

```
import datetime
import copy

from numpy import array
import pandas as pd

from keras.models import Sequential
from keras.layers import LSTM, Dense

def split_sequence(sequence, n_steps):
    X, y = list(), list()

    for i in range(len(sequence)):
        end_ix = i + n_steps
        if end_ix > len(sequence)-1:
            break
        seq_x, seq_y = sequence[i:end_ix], sequence[end_ix]
        X.append(seq_x)
        y.append(seq_y)

    return array(X), array(y)

df = pd.read_csv('2020-04-02 모니터링 데이터.csv')
df = df[df.GHID == 5].copy()[['MXSineMax'] + ['MTime']]
df['MTime'] = pd.to_datetime(df.MTime)
df = df.set_index(['MTime'])
# print(df.head())

raw_seq = list(df.iloc[:, 0])
n_step = 3

X, y = split_sequence(raw_seq, n_step)

# for i in range(len(X)):
#     print(X[i], y[i])

n_feature = 1
```

```
X = X.reshape((X.shape[0], X.shape[1], n_feature))
# print(X)

model = Sequential()
model.add(LSTM(50, activation='relu', input_shape=(n_step, n_feature)))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')

model.fit(X, y, epochs=200, verbose=0)

## 미완성
## x_input을 넣어 예측을 해야하는데, 완성하지 못함
# x_input = array()
# x_input = x_input.reshape(1, n_step, n_feature)

# y_hat = model.predict(x_input, verbose=0)
# print(y_hat)
```