Carga de librarias

In [1]:
```
# Librerias
import pandas as pd
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
import plotly.express as px

import seaborn as sns
from sklearn.preprocessing import OneHotEncoder, MinMaxScaler
from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score, confusion_matrix, plot_confusion

from keras.layers import Input, Dense, LSTM, TimeDistributed, RepeatVector
from keras.models import Model
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout, Bidirectional, Lambda
from keras.losses import MeanSquaredError
from keras.callbacks import EarlyStopping
from tensorflow.keras.optimizers import Adam, RMSprop
```

In [2]:
```
# Plot del training loss i l'accuracy
def plot_prediction(n_epochs, mfit):
    # TODO
    #Plots
    fig, ax = plt.subplots(nrows=2, ncols=1, figsize=(15,15))

    # plot accuracy during training
    ax[0].set_title('Mse')
    ax[0].plot(mfit.history['mse'], label='train')
    ax[0].plot(mfit.history['val_mse'], label='test')
    ax[0].legend()
    # plot loss during training
    ax[1].set_title('Loss')
    ax[1].plot(mfit.history['loss'], label='train')
    ax[1].plot(mfit.history['val_loss'], label='test')
    ax[1].legend()

def cut_cycles(df, lookback, future, column_features, column_label):
    df_feature = df[column_features]
    df_rul = df[column_label][lookback-1:]

    # Convertimos el dataframe en un numpy array
    numpy_features = df_feature.to_numpy()
    labels = df_rul.to_numpy()

    # Creación de listas vacías auxiliares
    features_set = []

    for i in range(lookback, df.shape[0] - future + 1):
        features_set.append(numpy_features[i - lookback:i])

    # Redimensionamiento numpy arrays
    features = np.array(features_set)
    features = np.reshape(features, (features.shape[0], features.shape[1], len(column_features) ))

    return features, labels

def modelo_lstm(input_shape, optimizer):
    model = Sequential()
    model.add(LSTM(units=100, input_shape=input_shape, return_sequences=True))
    model.add(Dropout(0.2))
    model.add(LSTM(units=30, return_sequences=False))
    model.add(Dropout(0.2))
    model.add(Dense(1, activation='relu'))
    model.add(Activation('relu'))
    model.compile(loss='mean_squared_error', optimizer=optimizer, metrics=['mse'])

    return model
```

Carga de datos

In [3]:
```
DATA_DIR = "C:/Users/NetRunner/OneDrive/UOC/Semestre 6/TFM/MultipleDatasets"

train_data = pd.read_csv(f"{DATA_DIR}/train_data.csv")
test_data = pd.read_csv(f"{DATA_DIR}/test_data.csv")
```

```
# X_train = pd.read_csv(f"{DATA_DIR}/X_train.csv")
# y_train = pd.read_csv(f"{DATA_DIR}/y_train.csv")
# X_test = pd.read_csv(f"{DATA_DIR}/X_test.csv")
# y_test = pd.read_csv(f"{DATA_DIR}/y_test.csv")
```

In [5]:

```
data = pd.concat([train_data, test_data])
features = ['volt','rotate','pressure','vibration','error1','error2','error3','error4','error5',
            'volt_3h_mean', 'rotate_3h_mean', 'pressure_3h_mean', 'vibration_3h_mean', 'volt_24h_mean',
            'rotate_24h_mean', 'pressure_24h_mean', 'vibration_24h_mean', 'error1_count', 'error2_count',
            'error3_count', 'error4_count', 'error5_count']
label = ['RUL']

print(len(train_data))
print(len(test_data))
print(len(data))
```

```
17280
3600
20880
```

In [6]:

```
feature_scaler = MinMaxScaler(feature_range=(0,1))
label_scaler = MinMaxScaler(feature_range=(0,1))

feature_scaler.fit(data[features])
label_scaler.fit(data[label].values.reshape(-1,1))
```

Out[6]:

```
MinMaxScaler()
```

In [7]:

```
data_norm = data[features+label].copy()
data_norm[features] = feature_scaler.transform(data[features])
data_norm[label] = label_scaler.transform(data[label].values.reshape(-1,1))
data_norm
```

Out[7]:

	volt	rotate	pressure	vibration	error1	error2	error3	error4	error5	volt_3h_mean	...	volt_24h_mean	rotate_24h_mean	pres
0	0.384246	0.635590	0.287496	0.550316	0.0	0.0	0.0	0.0	0.0	0.319571	...	0.370829	0.428252	
1	0.196252	0.595322	0.298321	0.255010	0.0	0.0	0.0	0.0	0.0	0.214811	...	0.341286	0.432249	
2	0.395865	0.468667	0.396966	0.388182	0.0	0.0	0.0	0.0	0.0	0.188577	...	0.332091	0.419061	
3	0.487156	0.700909	0.545746	0.547169	0.0	0.0	0.0	0.0	0.0	0.235855	...	0.322535	0.439945	
4	0.479313	0.592491	0.348040	0.421442	0.0	0.0	0.0	0.0	0.0	0.365896	...	0.326367	0.442053	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
3595	0.590798	0.456078	0.351645	0.786720	0.0	0.0	0.0	0.0	0.0	0.383354	...	0.375683	0.383491	
3596	0.455103	0.375885	0.410713	0.749031	0.0	0.0	0.0	0.0	0.0	0.414840	...	0.372778	0.371331	
3597	0.509347	0.491140	0.574390	0.500942	0.0	0.0	0.0	0.0	0.0	0.454524	...	0.366536	0.368975	
3598	0.638858	0.573666	0.258521	0.588043	0.0	0.0	0.0	0.0	0.0	0.476603	...	0.381726	0.371157	
3599	0.442910	0.657773	0.184867	0.551103	0.0	0.0	0.0	0.0	0.0	0.471001	...	0.371682	0.392093	

20880 rows × 23 columns

In [9]:

```
train_norm = data_norm[:len(train_data)]
test_norm = data_norm[len(train_data):(len(train_data)+len(test_data))]

X_train = train_norm.loc[:, train_norm.columns != 'RUL']
y_train = train_norm.loc[:, train_norm.columns == 'RUL']
X_test = test_norm.loc[:, test_norm.columns != 'RUL']
y_test = test_norm.loc[:, test_norm.columns == 'RUL']
```

Modelo LSTM

In [18]:

```
rangos = 72

train_3d = cut_cycles(train_norm, rangos, 0, features, label)

X_train_3d = train_3d[0]
y_train_3d = train_3d[1]


test_3d = cut_cycles(test_norm, rangos, 0, features, label)

X_test_3d = test_3d[0]
y_test_3d = test_3d[1]

print('X_train_3d:\t', X_train_3d.shape)
print('y_train_3d:\t', y_train_3d.shape)
```

```
print('X_test_3d:\t', X_test_3d.shape)
print('y_test_3d:\t', y_test_3d.shape)
```

```
X_train_3d:      (17209, 72, 22)
y_train_3d:      (17209, 1)
X_test_3d:       (3529, 72, 22)
y_test_3d:       (3529, 1)
```

In [11]:
```
%%time
epochs = [100]
batches = [8, 16]
optimizers = ['adam', 'rmsprop']
lrs = [0.01, 0.001, 0.0001]
input_shape=(X_train_3d.shape[1], X_train_3d.shape[2])

for epoch in epochs:
    for batch in batches:
        for opt in optimizers:
            for lr in lrs:
                if opt == 'adam':
                    optimizer = Adam(learning_rate=lr)
                elif opt == 'rmsprop':
                    optimizer = RMSprop(learning_rate=lr)

                model = modelo_lstm(input_shape, optimizer)

                print('=========Training model==========')

                print('Hiperparámetros:')
                print('Optimizer:\t', opt)
                print('Learning Rate:\t', lr)
                print('Epochs:\t\t', epoch)
                print('Batch:\t\t', batch)

                res_m = model.fit(X_train_3d, y_train_3d, validation_data=(X_test_3d, y_test_3d), epochs=epoch,

                pred = model.predict(X_test_3d)
                pred
                pred_list = [x[0] for x in pred]



                print('\nEvaluation:')
                print('R^2 score:\t\t', r2_score(y_test_3d, pred_list))
                print('MSE score:\t\t', mean_squared_error(y_test_3d, pred_list))
                print('MAE score:\t\t', mean_absolute_error(y_test_3d, pred_list))

                print('=================================\n')
```

```
=========Training model==========
Hiperparámetros:
Optimizer:       adam
Learning Rate:   0.01
Epochs:          100
Batch:           8
Restoring model weights from the end of the best epoch: 4.
Epoch 14: early stopping
111/111 [==============================] - 2s 13ms/step

Evaluation:
R^2 score:               0.45022060900476635
MSE score:               0.044559361514754245
MAE score:               0.16651966790417883
=================================

=========Training model==========
Hiperparámetros:
Optimizer:       adam
Learning Rate:   0.001
Epochs:          100
Batch:           8
Restoring model weights from the end of the best epoch: 9.
Epoch 19: early stopping
111/111 [==============================] - 2s 13ms/step

Evaluation:
R^2 score:               0.5060367010255538
MSE score:               0.04003549346252924
MAE score:               0.1536788379250833
=================================

=========Training model==========
Hiperparámetros:
Optimizer:       adam
Learning Rate:   0.0001
Epochs:          100
Batch:           8
Restoring model weights from the end of the best epoch: 5.
Epoch 15: early stopping
```

```
111/111 [==============================] - 2s 14ms/step

Evaluation:
R^2 score:               0.426638400657786
MSE score:               0.04647068842925923
MAE score:               0.17323612724829365
=================================

=========Training model==========
Hiperparámetros:
Optimizer:       rmsprop
Learning Rate:   0.01
Epochs:          100
Batch:           8
Restoring model weights from the end of the best epoch: 1.
Epoch 11: early stopping
111/111 [==============================] - 2s 13ms/step

Evaluation:
R^2 score:               -2.97351888864436
MSE score:               0.322051840328703
MAE score:               0.49091986091005935
=================================

=========Training model==========
Hiperparámetros:
Optimizer:       rmsprop
Learning Rate:   0.001
Epochs:          100
Batch:           8
Restoring model weights from the end of the best epoch: 3.
Epoch 13: early stopping
111/111 [==============================] - 2s 15ms/step

Evaluation:
R^2 score:               0.41004842703113575
MSE score:               0.04781529800258636
MAE score:               0.17945221024350452
=================================

=========Training model==========
Hiperparámetros:
Optimizer:       rmsprop
Learning Rate:   0.0001
Epochs:          100
Batch:           8
Restoring model weights from the end of the best epoch: 7.
Epoch 17: early stopping
111/111 [==============================] - 2s 15ms/step

Evaluation:
R^2 score:               0.34927634454760537
MSE score:               0.052740846754943234
MAE score:               0.18829420043431302
=================================

=========Training model==========
Hiperparámetros:
Optimizer:       adam
Learning Rate:   0.01
Epochs:          100
Batch:           16
Restoring model weights from the end of the best epoch: 3.
Epoch 13: early stopping
111/111 [==============================] - 2s 13ms/step

Evaluation:
R^2 score:               0.2899622072977417
MSE score:               0.057548229730627266
MAE score:               0.19129440954535235
=================================

=========Training model==========
Hiperparámetros:
Optimizer:       adam
Learning Rate:   0.001
Epochs:          100
Batch:           16
Restoring model weights from the end of the best epoch: 10.
Epoch 20: early stopping
111/111 [==============================] - 2s 16ms/step

Evaluation:
R^2 score:               0.3101378682429161
MSE score:               0.05591300188363995
MAE score:               0.18090818622730068
=================================

=========Training model==========
```

```
Hiperparámetros:
Optimizer:       adam
Learning Rate:   0.0001
Epochs:          100
Batch:           16
Restoring model weights from the end of the best epoch: 3.
Epoch 13: early stopping
111/111 [==============================] - 2s 16ms/step

Evaluation:
R^2 score:                0.2570221060564791
MSE score:                0.06021800947061528
MAE score:                0.20239441684380324
=================================

=========Training model==========
Hiperparámetros:
Optimizer:       rmsprop
Learning Rate:   0.01
Epochs:          100
Batch:           16
Restoring model weights from the end of the best epoch: 1.
Epoch 11: early stopping
111/111 [==============================] - 2s 14ms/step

Evaluation:
R^2 score:                -2.97351888864436
MSE score:                0.322051840328703
MAE score:                0.49091986091005935
=================================

=========Training model==========
Hiperparámetros:
Optimizer:       rmsprop
Learning Rate:   0.001
Epochs:          100
Batch:           16
Restoring model weights from the end of the best epoch: 10.
Epoch 20: early stopping
111/111 [==============================] - 2s 16ms/step

Evaluation:
R^2 score:                0.4041971966084157
MSE score:                0.048289537481152715
MAE score:                0.16687301351377531
=================================

=========Training model==========
Hiperparámetros:
Optimizer:       rmsprop
Learning Rate:   0.0001
Epochs:          100
Batch:           16
Restoring model weights from the end of the best epoch: 6.
Epoch 16: early stopping
111/111 [==============================] - 2s 16ms/step

Evaluation:
R^2 score:                0.4470208276119697
MSE score:                0.044818702294319855
MAE score:                0.164409799681083
=================================

CPU times: total: 8h 23min 12s
Wall time: 2h 28min 24s
```

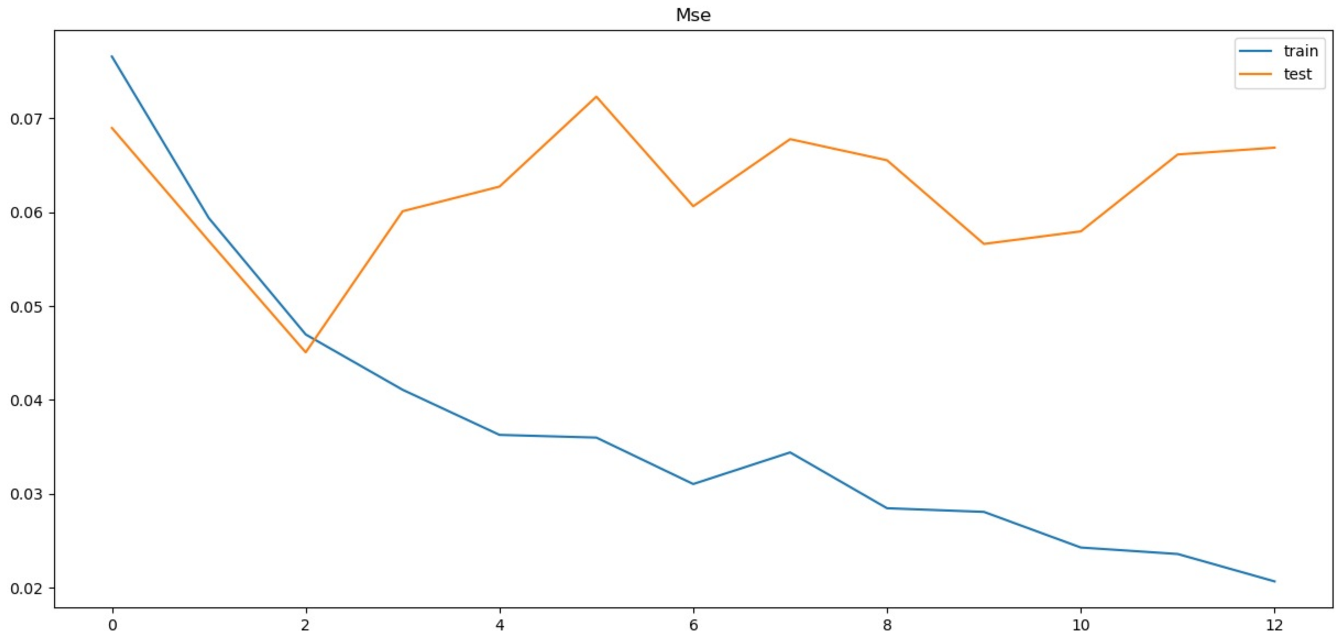
In [19]:

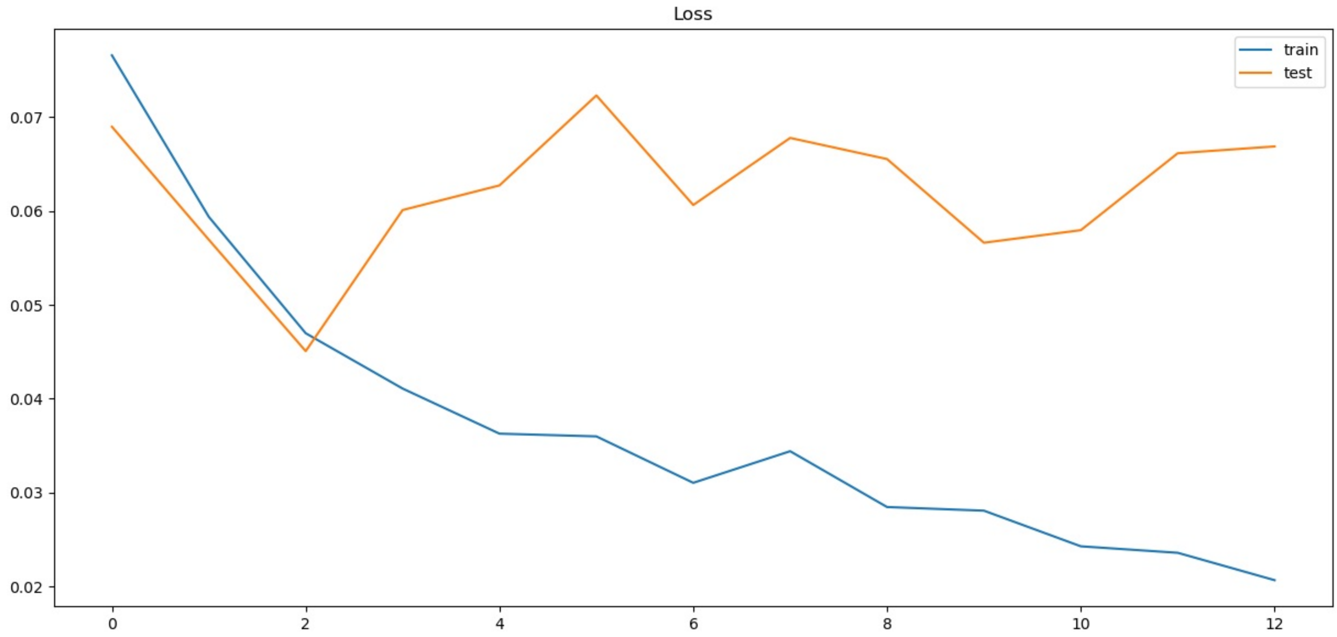
```python
epochs = 50
optimizer = Adam(learning_rate=0.001)
input_shape=(X_train_3d.shape[1], X_train_3d.shape[2])

model = modelo_lstm(input_shape, optimizer)

res_m = model.fit(X_train_3d, y_train_3d, validation_data=(X_test_3d, y_test_3d), epochs=epochs, batch_size=8,
```

```
Epoch 1/50
2152/2152 [==============================] - 116s 53ms/step - loss: 0.0766 - mse: 0.0766 - val_loss: 0.0689 - v
al_mse: 0.0689
Epoch 2/50
2152/2152 [==============================] - 125s 58ms/step - loss: 0.0594 - mse: 0.0594 - val_loss: 0.0569 - v
al_mse: 0.0569
Epoch 3/50
2152/2152 [==============================] - 117s 54ms/step - loss: 0.0470 - mse: 0.0470 - val_loss: 0.0451 - v
al_mse: 0.0451
Epoch 4/50
2152/2152 [==============================] - 112s 52ms/step - loss: 0.0411 - mse: 0.0411 - val_loss: 0.0601 - v
al_mse: 0.0601
Epoch 5/50
2152/2152 [==============================] - 113s 52ms/step - loss: 0.0363 - mse: 0.0363 - val_loss: 0.0627 - v
al_mse: 0.0627
Epoch 6/50
2152/2152 [==============================] - 112s 52ms/step - loss: 0.0360 - mse: 0.0360 - val_loss: 0.0723 - v
al_mse: 0.0723
Epoch 7/50
2152/2152 [==============================] - 113s 52ms/step - loss: 0.0311 - mse: 0.0311 - val_loss: 0.0606 - v
al_mse: 0.0606
Epoch 8/50
2152/2152 [==============================] - 114s 53ms/step - loss: 0.0344 - mse: 0.0344 - val_loss: 0.0678 - v
al_mse: 0.0678
Epoch 9/50
2152/2152 [==============================] - 116s 54ms/step - loss: 0.0285 - mse: 0.0285 - val_loss: 0.0655 - v
al_mse: 0.0655
Epoch 10/50
2152/2152 [==============================] - 118s 55ms/step - loss: 0.0281 - mse: 0.0281 - val_loss: 0.0566 - v
al_mse: 0.0566
Epoch 11/50
2152/2152 [==============================] - 113s 52ms/step - loss: 0.0243 - mse: 0.0243 - val_loss: 0.0579 - v
al_mse: 0.0579
Epoch 12/50
2152/2152 [==============================] - 111s 51ms/step - loss: 0.0236 - mse: 0.0236 - val_loss: 0.0661 - v
al_mse: 0.0661
Epoch 13/50
2152/2152 [==============================] - ETA: 0s - loss: 0.0207 - mse: 0.0207Restoring model weights from t
he end of the best epoch: 3.
2152/2152 [==============================] - 111s 51ms/step - loss: 0.0207 - mse: 0.0207 - val_loss: 0.0669 - v
al_mse: 0.0669
Epoch 13: early stopping
```

In [20]:

```
plot_prediction(epochs, res_m)
```





In [22]:

```
y_test_3d_5 = y_test_3d[(1*720)-rangos+1:(2*720)-rangos+1]
pred = model.predict(X_test_3d[(1*720)-rangos+1:(2*720)-rangos+1])
temp_list = []
for i in range(pred.shape[0]):
    temp_list.append(pred[i])


plt.figure(figsize=(15, 10))
plt.plot(temp_list)
plt.plot(y_test_3d_5)
plt.show()
```

```
23/23 [==============================] - 1s 26ms/step
```

1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400
500
600
700