In [2]:
```
import os
print("Current working directory:", os.getcwd())
```

```
Current working directory: c:\Work\MS-AI\course-7\Project\IoT_Gas_Turbine\notebooks
```

In [3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import glob
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense
```

In [4]:
```
# Load & combine Data CSV files

train_files = glob.glob('../dataset/train/*.csv')
train_data = pd.concat([pd.read_csv(f) for f in train_files], ignore_index=True)
print("Total training rows", {train_data.shape[0]})
```

```
Total training rows {52940}
```

In [5]:
```
# Data Cleaning

train_data = train_data.drop_duplicates()
train_data = train_data.dropna()

print("Total training rows", {train_data.shape[0]})
train_data.head()
```

```
Total training rows {52940}
```

Out[5]:

	time	input_voltage	el_power
0	810.07028	10.0	1228.791720
1	811.06938	10.0	1223.041745
2	812.06848	10.0	1244.960866
3	813.06758	10.0	1229.259058
4	814.06668	10.0	1248.117024

In [6]:
```
# Ensure numeric types
```

```
train_data["input_voltage"] = pd.to_numeric(train_data["input_voltage"], errors='coerce')
train_data["el_power"] = pd.to_numeric(train_data["el_power"], errors='coerce')
train_data.dropna()
print(f"Training rows after cleaning: {train_data.shape[0]}", train_data.head())
```

```
Training rows after cleaning: 52940          time  input_voltage     el_power
0  810.07028           10.0  1228.791720
1  811.06938           10.0  1223.041745
2  812.06848           10.0  1244.960866
3  813.06758           10.0  1229.259058
4  814.06668           10.0  1248.117024
```

In [7]:
```
# Feature Selection

X = train_data[['input_voltage']]
y = train_data['el_power']

print(f"X shape: {X.shape}")
print(f"y shape: {y.shape}")
```

```
X shape: (52940, 1)
y shape: (52940,)
```

In [8]:
```
# Training Validation split

split_idx = int(len(X) * 0.8)

X_train = X.iloc[:split_idx]
X_val   = X.iloc[split_idx:]

y_train = y.iloc[:split_idx]
y_val   = y.iloc[split_idx:]

print(f"Training samples: {X_train.shape[0]}")
print(f"Validation samples: {X_val.shape[0]}")
```

```
Training samples: 42352
Validation samples: 10588
```

In [9]:
```
# Feature scaling

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_val_scaled = scaler.transform(X_val)
```

In [10]:
```
# Build deep learning model

model = Sequential([
    Dense(64, activation='relu', input_shape=(X_train_scaled.shape[1],)),
    Dense(64, activation='relu'),
    Dense(32, activation='relu'),
    Dense(1) # Output Layer: predicts el_power

])

model.compile(optimizer='adam', loss='mse', metrics=['mae'])
model.summary()
```

```
c:\Work\MS-AI\course-7\Project\IoT_Gas_Turbine\course7_Project_venv\lib\site-packages\keras\src\layers\core\dense.py:95: UserWarnin
g: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` objec
t as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 64)	128
dense_1 (Dense)	(None, 64)	4,160
dense_2 (Dense)	(None, 32)	2,080
dense_3 (Dense)	(None, 1)	33

Total params: 6,401 (25.00 KB)

Trainable params: 6,401 (25.00 KB)

Non-trainable params: 0 (0.00 B)

In [11]:
```
history = model.fit(
    X_train_scaled,
    y_train,
    validation_data=(X_val_scaled, y_val),
    epochs=50,
    batch_size=32,
    verbose=1
)
```

```
Epoch 1/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - loss: 658593.8125 - mae: 458.3993 - val_loss: 21352.9863 - val_mae: 115.4910
Epoch 2/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 140745.3281 - mae: 217.1256 - val_loss: 10286.0498 - val_mae: 81.7886
Epoch 3/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 139790.9531 - mae: 215.0523 - val_loss: 10369.5098 - val_mae: 81.7505
Epoch 4/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 139639.1406 - mae: 214.7768 - val_loss: 14340.7422 - val_mae: 96.9883
Epoch 5/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 2s 2ms/step - loss: 139444.4375 - mae: 213.9353 - val_loss: 10801.5645 - val_mae: 83.7251
Epoch 6/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 139509.4531 - mae: 214.3458 - val_loss: 7830.9800 - val_mae: 72.6539
Epoch 7/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 139355.2188 - mae: 214.2080 - val_loss: 9384.3086 - val_mae: 78.6064
Epoch 8/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 139288.2344 - mae: 213.5420 - val_loss: 12017.8672 - val_mae: 88.1223
Epoch 9/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 139374.2188 - mae: 213.9189 - val_loss: 18860.3945 - val_mae: 111.3012
Epoch 10/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 139288.9688 - mae: 213.3970 - val_loss: 15063.8564 - val_mae: 98.6328
Epoch 11/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 139201.1406 - mae: 213.3337 - val_loss: 14388.6396 - val_mae: 96.4909
Epoch 12/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 139159.5312 - mae: 213.3777 - val_loss: 15431.4590 - val_mae: 99.9184
Epoch 13/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 139102.6250 - mae: 213.3388 - val_loss: 16954.3359 - val_mae: 105.0978
Epoch 14/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 139051.8281 - mae: 213.4538 - val_loss: 14770.7305 - val_mae: 98.4125
Epoch 15/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 138970.4062 - mae: 212.6339 - val_loss: 11865.8643 - val_mae: 87.5174
Epoch 16/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - loss: 139024.2656 - mae: 213.1413 - val_loss: 19002.0859 - val_mae: 111.3638
Epoch 17/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - loss: 138976.2969 - mae: 213.0689 - val_loss: 12113.0674 - val_mae: 88.9422
Epoch 18/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 138807.0938 - mae: 212.6702 - val_loss: 15577.4023 - val_mae: 101.4665
Epoch 19/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - loss: 139031.1094 - mae: 212.6071 - val_loss: 14248.9502 - val_mae: 95.5600
Epoch 20/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - loss: 138837.5156 - mae: 212.8958 - val_loss: 9665.7227 - val_mae: 79.1592
Epoch 21/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 138939.6406 - mae: 212.6091 - val_loss: 12275.5664 - val_mae: 89.0167
Epoch 22/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 138784.1406 - mae: 212.4341 - val_loss: 11895.1152 - val_mae: 87.1441
Epoch 23/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 138892.8438 - mae: 212.5867 - val_loss: 9608.6992 - val_mae: 78.7191
Epoch 24/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 138598.2812 - mae: 212.4905 - val_loss: 13555.3076 - val_mae: 94.1641
```

```
Epoch 25/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 138671.9688 - mae: 212.3960 - val_loss: 8834.0986 - val_mae: 76.1982
Epoch 26/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 138733.2031 - mae: 212.1030 - val_loss: 11720.7861 - val_mae: 86.3077
Epoch 27/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 138625.5781 - mae: 212.2124 - val_loss: 10924.8145 - val_mae: 83.2903
Epoch 28/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 138583.5312 - mae: 212.0393 - val_loss: 15444.9961 - val_mae: 99.5554
Epoch 29/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 138590.0156 - mae: 212.4370 - val_loss: 13273.6289 - val_mae: 93.2087
Epoch 30/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 138539.0625 - mae: 212.0290 - val_loss: 13438.0400 - val_mae: 93.5779
Epoch 31/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 138485.6094 - mae: 212.3255 - val_loss: 11376.2188 - val_mae: 85.6503
Epoch 32/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - loss: 138524.8594 - mae: 212.0735 - val_loss: 17877.5586 - val_mae: 109.8689
Epoch 33/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - loss: 138492.8594 - mae: 211.8269 - val_loss: 16255.9941 - val_mae: 103.4973
Epoch 34/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 138511.1094 - mae: 212.2982 - val_loss: 14795.0342 - val_mae: 98.8104
Epoch 35/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - loss: 138600.9219 - mae: 212.2158 - val_loss: 13352.0420 - val_mae: 93.7598
Epoch 36/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 138698.0781 - mae: 212.4054 - val_loss: 14658.0723 - val_mae: 98.8511
Epoch 37/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - loss: 138749.0781 - mae: 212.3846 - val_loss: 14050.5283 - val_mae: 96.4699
Epoch 38/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 138521.9844 - mae: 212.7088 - val_loss: 10935.5576 - val_mae: 85.1184
Epoch 39/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 138479.4062 - mae: 211.8815 - val_loss: 16604.5801 - val_mae: 105.2597
Epoch 40/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 138342.2500 - mae: 212.0362 - val_loss: 12422.2607 - val_mae: 90.5721
Epoch 41/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - loss: 138501.7656 - mae: 212.1303 - val_loss: 13094.0732 - val_mae: 93.0693
Epoch 42/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 138442.8594 - mae: 212.2153 - val_loss: 9819.6035 - val_mae: 80.1005
Epoch 43/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 138561.2188 - mae: 212.1378 - val_loss: 11874.1318 - val_mae: 88.6235
Epoch 44/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 138368.2656 - mae: 212.2772 - val_loss: 13057.2568 - val_mae: 92.9250
Epoch 45/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 138429.3594 - mae: 212.1671 - val_loss: 12570.3926 - val_mae: 91.0569
Epoch 46/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 138585.0312 - mae: 212.4183 - val_loss: 11265.1924 - val_mae: 86.1154
Epoch 47/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - loss: 138562.3125 - mae: 212.3159 - val_loss: 12419.9150 - val_mae: 90.5108
Epoch 48/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 138461.8281 - mae: 211.9419 - val_loss: 9984.3047 - val_mae: 80.9785
```

```
Epoch 49/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - loss: 138392.0469 - mae: 212.1057 - val_loss: 10137.5127 - val_mae: 81.4633
Epoch 50/50
1324/1324 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - loss: 138513.0156 - mae: 212.2353 - val_loss: 13854.9668 - val_mae: 95.9970
```

In [12]:

```
val_loss, val_mae = model.evaluate(X_val_scaled, y_val, verbose=0)
print(f"\nValidation MSE: {val_loss:.3f}, MAE: {val_mae:.3f}")
```

```
Validation MSE: 13854.967, MAE: 95.997
```

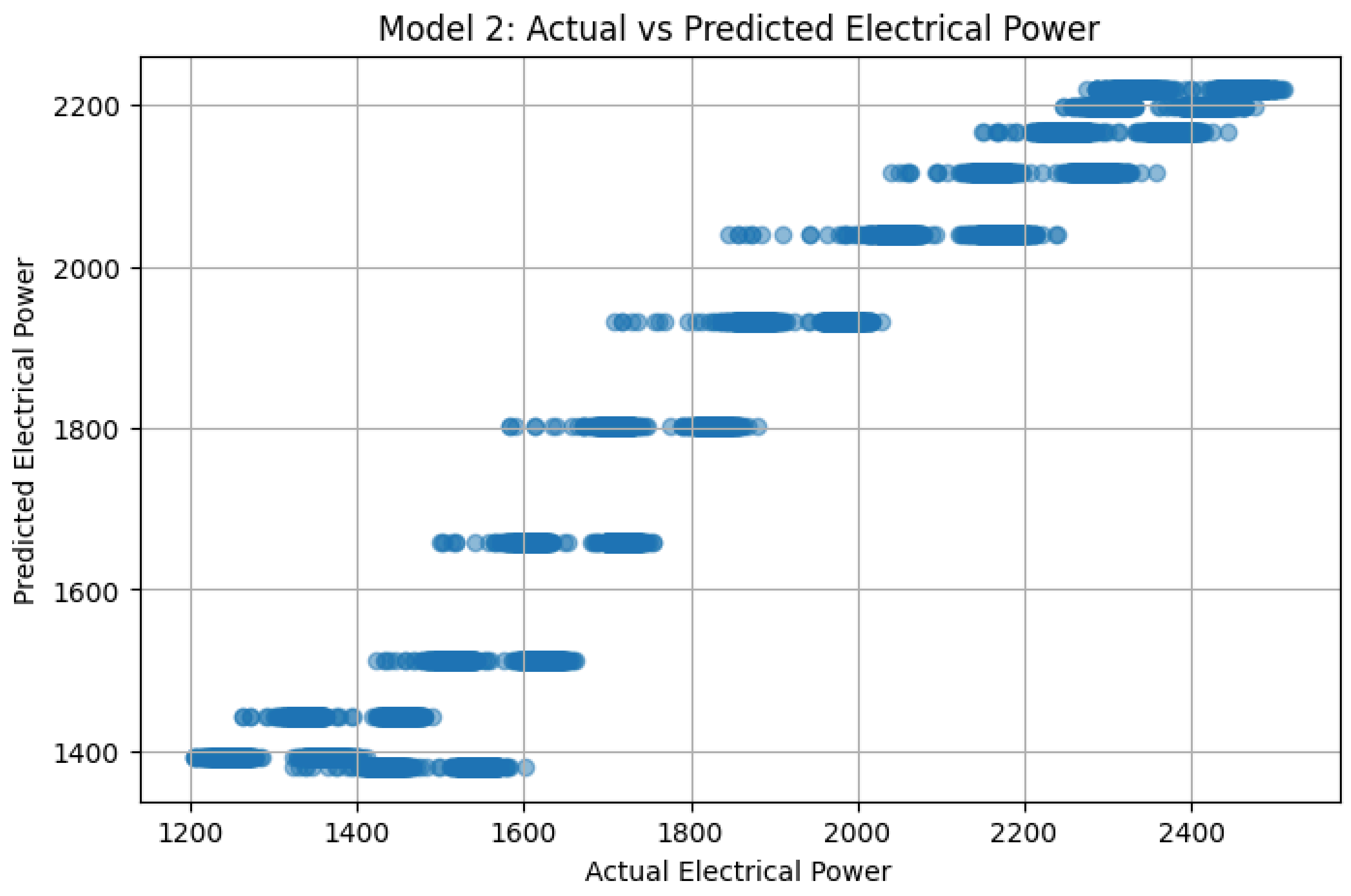
In [13]:

```
# Validation Visualization

y_val_pred = model.predict(X_val_scaled)

plt.figure(figsize=(8,5))
plt.scatter(y_val, y_val_pred, alpha=0.5)
plt.xlabel("Actual Electrical Power")
plt.ylabel("Predicted Electrical Power")
plt.title("Model 2: Actual vs Predicted Electrical Power")
plt.grid(True)
plt.show()
```

```
331/331 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
```



In [14]:
```python
# Make Prediction of Test data

test_files = glob.glob("../dataset/test/*.csv")

for f in test_files:
    test_data = pd.read_csv(f)

    test_data["input_voltage"] = pd.to_numeric(test_data["input_voltage"], errors='coerce')
    test_data["el_power"] = pd.to_numeric(test_data["el_power"], errors='coerce')

    test_data = test_data.dropna().drop_duplicates()
    test_data = test_data.sort_values(by="time")

    X_test_scaled = scaler.transform(test_data[['input_voltage']])
    test_data['predicted_el_power'] = model.predict(X_test_scaled)

    output_file = f.replace(".csv", "_dnn_predictions.csv")
    test_data.to_csv(output_file, index=False)
    print(f"Saved predictions to {output_file}")
```

```
266/266 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
Saved predictions to ../dataset/test\ex_22_dnn_predictions.csv
266/266 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
Saved predictions to ../dataset/test\ex_22_dnn_predictions_dnn_predictions.csv
307/307 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
Saved predictions to ../dataset/test\ex_4_dnn_predictions.csv
307/307 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
Saved predictions to ../dataset/test\ex_4_dnn_predictions_dnn_predictions.csv
```