**GOVERNMENT COLLEGE OF ENGINEERING ERODE**



B.E Electronics and Communication Engineering

**Name of the Students: University Register no:**

**Team Leader:**

Anburaj.S 731121106103

**Team Members:**

Dharaneeswaran.K 731121106008

Thanush Vishal.S 731121106051

Boopathiraj.V 731121106003

Prabhakaran.S 731121106035

Under the mentor of

**Dr.M.Poongothai**

**Department of Information Technology(IT)**

**Department of Electronics and Communication Engineering**

Government College of Engineering

Erode ,PO ,near Vasavi College,TamilNadu-638316,

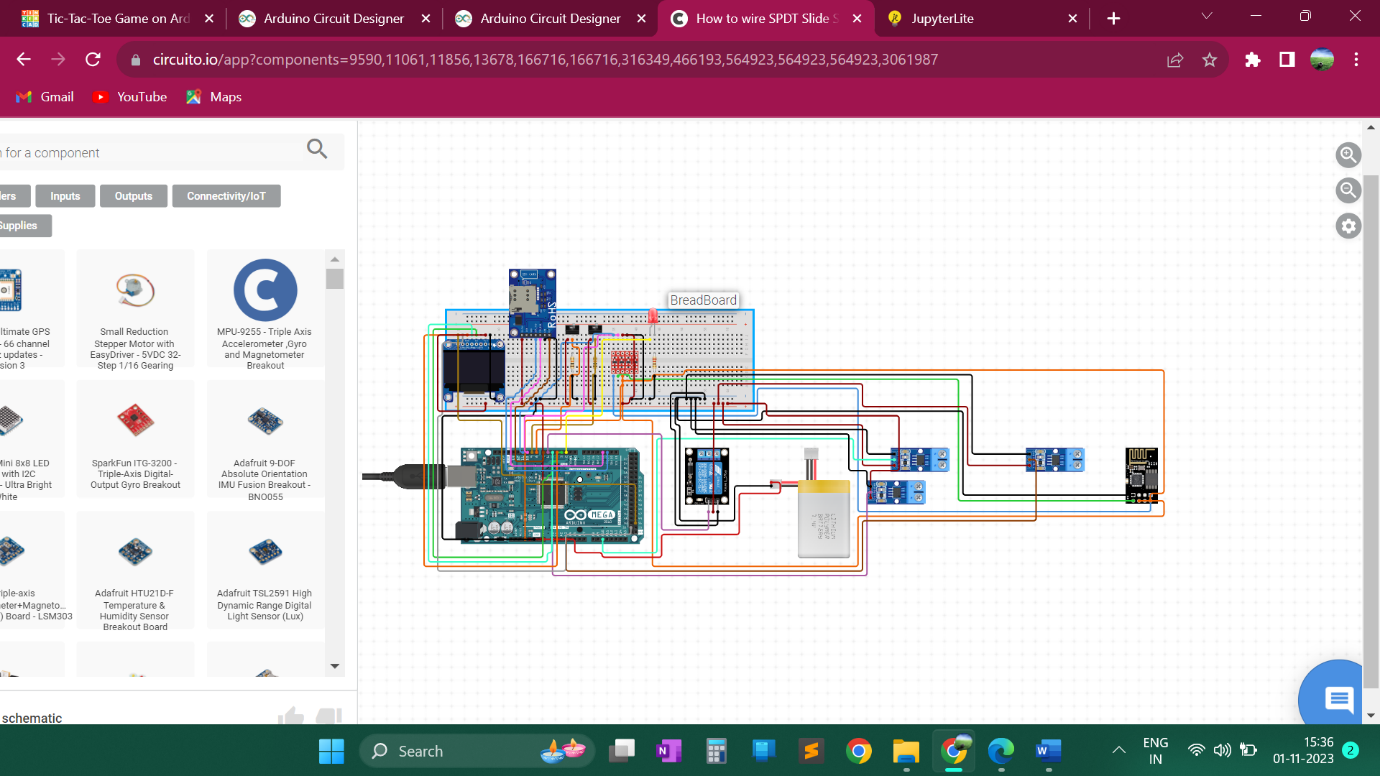
Affiliated to Anna University ,Chennai.

**MEASURE ENERGY CONSUMPTION**

**INTRODUCTION:**

Energy consumption has been widely studied in the [computer architecture](https://www.sciencedirect.com/topics/computer-science/computer-architecture) field for decades. While the adoption of energy as a metric in [machine learning](https://www.sciencedirect.com/topics/computer-science/machine-learning) is emerging, the majority of research is still primarily focused on obtaining high levels of accuracy without any computational constraint. We believe that one of the reasons for this lack of interest is due to their lack of familiarity with approaches to evaluate energy consumption.

**3D MODEL CIRCUIT:**

****

**PROGRAM:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import warnings

warnings.filterwarnings("ignore", category=UserWarning)

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.svm import SVR

from sklearn.metrics import mean\_squared\_error, r2\_score

RED = "\033[91m"

GREEN = "\033[92m"

YELLOW = "\033[93m"

BLUE = "\033[94m"

RESET = "\033[0m"

df = pd.read\_csv(r'C:\Users\user\Desktop\AEP\_hourly.csv')

df["Datetime"] = pd.to\_datetime(df["Datetime"])

# DATA CLEANING

print(BLUE + "\nDATA CLEANING" + RESET)

# --- Check for missing values

missing\_values = df.isnull().sum()

print(GREEN + "Missing Values : " + RESET)

print(missing\_values)

# --- Handle missing values

df.dropna(inplace=True)

# --- Check for duplicate values

duplicate\_values = df.duplicated().sum()

print(GREEN + "Duplicate Values : " + RESET)

print(duplicate\_values)

# --- Drop duplicate values

df.drop\_duplicates(inplace=True)

# DATA ANALYSIS

print(BLUE + "\nDATA ANALYSIS" + RESET)

# --- Summary Statistics

summary\_stats = df.describe()

print(GREEN + "Summary Statistics : " + RESET)

print(summary\_stats)

# SUPPORT VECTOR MODELLLING

print(BLUE + "\nMODELLING" + RESET)

# Reduce the dataset size for faster training

df = df.sample(frac=0.2, random\_state=42)

# Split the data into features (Datetime) and target (AEP\_MW)

X = df[["Datetime"]]

y = df["AEP\_MW"]

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y, test\_size=0.2, random\_state=42

)

# Preprocess the features (Datetime) to extract the day of the year

X\_train["DayOfYear"] = X\_train["Datetime"].dt.dayofyear

X\_test["DayOfYear"] = X\_test["Datetime"].dt.dayofyear

# Convert X\_train and X\_test to NumPy arrays

X\_train = X\_train["DayOfYear"].values.reshape(-1, 1)

X\_test = X\_test["DayOfYear"].values.reshape(-1, 1)

# Standardize the data

scaler = StandardScaler()

X\_train\_scaled = scaler.fit\_transform(X\_train)

X\_test\_scaled = scaler.transform(X\_test)

# Create an SVR (Support Vector Regression) model with a linear kernel

svr = SVR(kernel="linear", C=1.0)

# Train the SVR model

svr.fit(X\_train\_scaled, y\_train)

# Predict on the test set

y\_pred = svr.predict(X\_test\_scaled)

# Evaluate the model

mse = mean\_squared\_error(y\_test, y\_pred)

r2 = r2\_score(y\_test, y\_pred)

print(f"Mean Squared Error: {mse}")

print(f"R-squared: {r2}")

# Plot the actual vs. predicted values

plt.figure(figsize=(10, 6))

plt.scatter(X\_test, y\_test, color="b", label="Actual")

plt.scatter(X\_test, y\_pred, color="r", label="Predicted")

plt.xlabel("Day of the Year")

plt.ylabel("Energy Consumption (MW)")

plt.title("SVR Model: Actual vs. Predicted")

plt.legend()

plt.grid()

plt.show()

# DATA VISUALIZATION

print(BLUE + "\nDATA VISUALIZATION" + RESET)

# --- Line plot

print(GREEN + "LinePlot : " + RESET)

plt.figure(figsize=(10, 6))

sns.lineplot(data=df, x="Datetime", y="AEP\_MW")

plt.xlabel("Datetime")

plt.ylabel("Energy Consumption (MW)")

plt.title("Energy Consumption Over Year")

plt.grid()

plt.show()

# --- Histogram

print(GREEN + "Histogram : " + RESET)

plt.figure(figsize=(10, 6))

plt.hist(

df["AEP\_MW"],

bins=100,

histtype="barstacked",

edgecolor="white",

)

plt.xlabel("AEPMW")

plt.ylabel("Frequency")

plt.title("Histogram of MEGAWATT USAGE")

plt.show()

**OUTPUT:**

DATA CLEANING

Missing Values :

Datetime 0

AEP\_MW 0

dtype: int64

Duplicate Values :

0

DATA ANALYSIS

Summary Statistics :

Datetime AEP\_MW

count 121273 121273.000000

mean 2011-09-02 03:17:01.553025024 15499.513717

min 2004-10-01 01:00:00 9581.000000

25% 2008-03-17 15:00:00 13630.000000

50% 2011-09-02 04:00:00 15310.000000

75% 2015-02-16 17:00:00 17200.000000

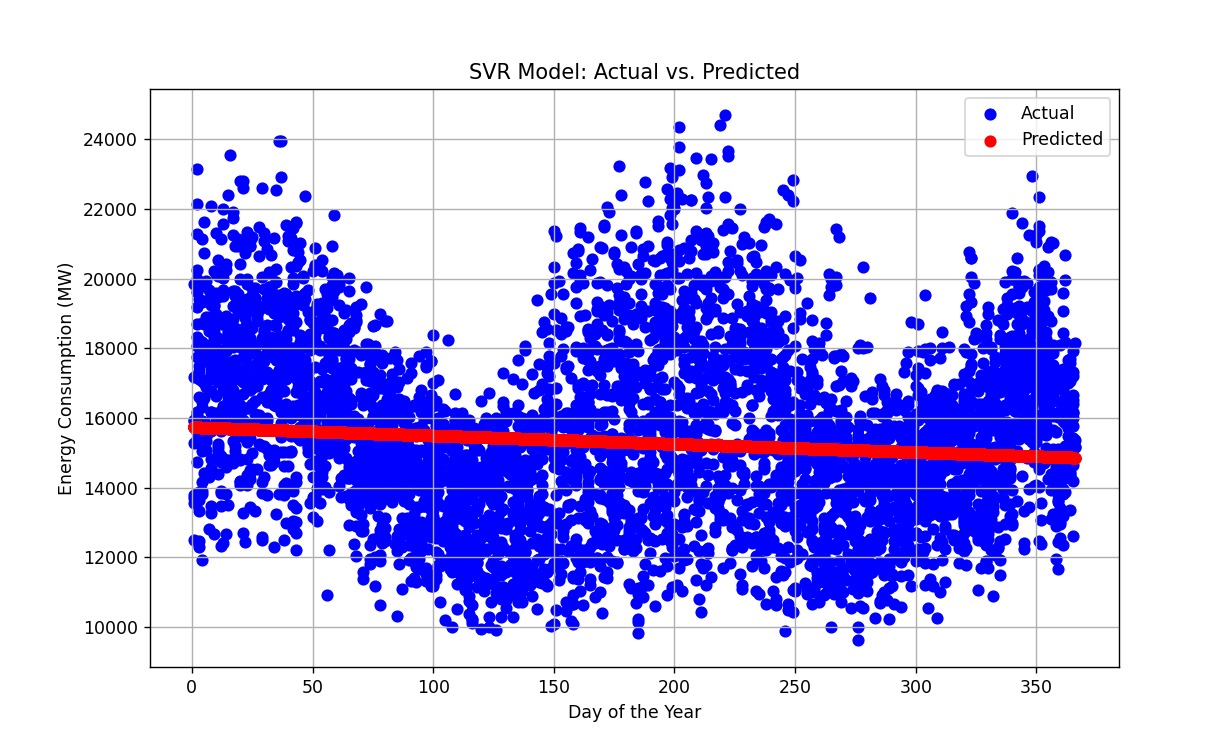
max 2018-08-03 00:00:00 25695.000000

std NaN 2591.399065

**MODELLING:**

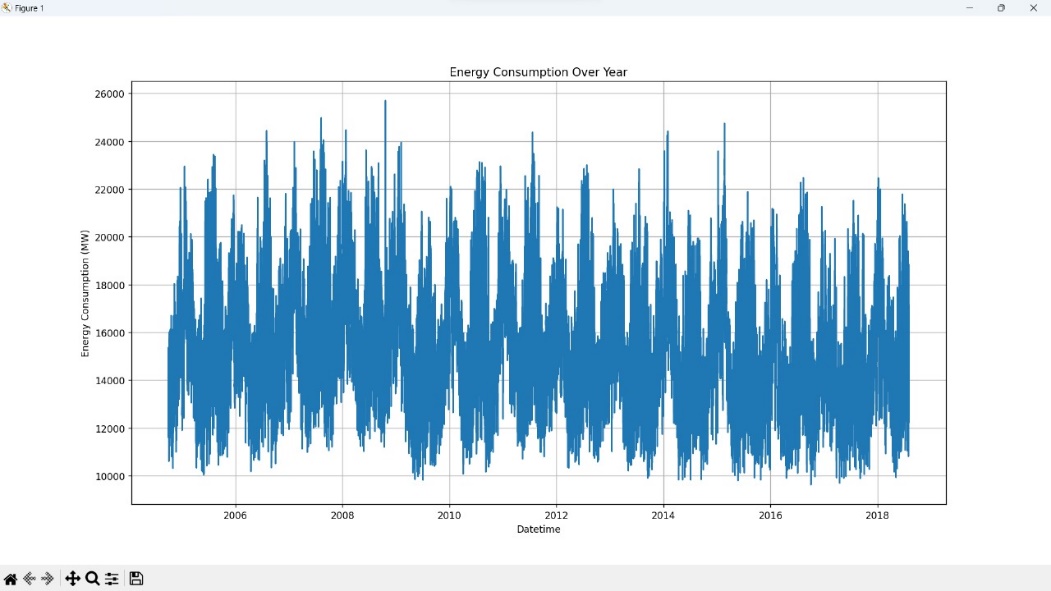
Mean Squared Error: 6758395.805638685

R-squared: 0.00270160624748228



**DATA VISUALIZATION:**

LinePlot:



Histogram:

