

MEASURE ENERGY CONSUMPTION

PHASE 5 SUBMISSION DOCUMENT

PHASE 5: PROJECT DOCUMENTATION & SUBMISSION



INTRODUCTION:

Design thinking is an iterative process that can be applied to solve complex problems, such as measuring energy consumption. In this document, we will outline a design thinking approach for developing a solution to measure and manage energy consumption effectively.

Measuring energy consumption is a key component efforts, allowing for the reduction of carbon footprints and environmental impact.

Many regulatory and environmental standards measurement and reporting of energy consumption compliance with energy efficiency and emission targets.

TOOLS AND SOFTWARE COMMONLY USED IN THE PROCESS:

1. Power Meters:

Wattmeters: These physical devices can be connected between the power source and the computer to measure real-time power consumption.

Smart Plugs: Some smart plugs can monitor energy consumption and provide data through apps or APIs.

2. Energy Measurement Hardware:

Energy Monitoring Boards: These are specialized hardware components that can be integrated into a computer system to measure energy consumption.

3. Operating System Utilities:

Windows Task Manager (Windows): On Windows systems, you can use the Task Manager to monitor energy consumption.

Activity Monitor (macOS): In macOS, the Activity Monitor provides energy usage statistics.

4. Power Monitoring Libraries:

Intel Power Gadget: This tool is useful for measuring power consumption on Intel-based systems.

RAPL (Running Average Power Limit): It's a feature available on some Intel processors for monitoring power usage.

ACPI (Advanced Configuration and Power Interface): ACPI provides information about power management on many systems.

5. Energy Profiling Tools for Mobile Devices:

Battery Historian (Android): It's a tool for analyzing battery consumption on Android devices.

Instruments (iOS): For iOS development, Instruments can be used to profile energy usage.

6.System Profilers:

Intel VTune Profiler: This tool can help measure power and energy consumption for CPU-bound tasks.

NVIDIA NVML (NVIDIA Management Library): If you're using NVIDIA GPUs, you can use NVML to monitor power usage.

7.Software Libraries:

PSUtil (Python): As mentioned earlier, `psutil` is a Python library that provides an interface to monitor system-related information, including battery status and power usage.

8. Power Measurement Tools in Development Frameworks:

Some machine learning development frameworks (e.g., TensorFlow, PyTorch) provide tools for measuring energy consumption during model training.

9. Specialized Energy Measurement Tools:

Some companies and research institutions develop specialized hardware and software for energy measurement, tailored to their specific needs.

It's essential to choose the right tool or combination of tools based on your hardware, operating system, and specific use case. The accuracy and granularity of energy consumption measurements can vary depending on the tools and hardware you use.

Here is a general process for measuring energy consumption in a software process:

1. Identify the Target Process:

Determine which software process or application you want to measure the energy consumption of. This could be a specific program, service, or a set of tasks within a larger application.

2. Select Appropriate Energy Monitoring Tools:

Choose the right tools and software for measuring energy consumption. These tools can vary depending on the operating system, platform, and the level of granularity you require. Some common tools and steps include:

OS-Level Tools: Many operating systems provide built-in tools for monitoring energy consumption. For example:

Windows: Use the "Power Usage" section in Task Manager.

macOS: Use the "Energy" tab in Activity Monitor.

Linux: Various command-line tools like `powertop` and `acpi` can provide power consumption information.

Profiling Tools: Software profiling tools can help you measure CPU and memory usage, which indirectly affects energy consumption. Examples include:

Intel VTune Profile: Offers insights into CPU usage and power consumption.

Xcode Instruments (macOS/iOS): Provides energy profiling for macOS and iOS applications.

Android Profiler (Android Studio): Helps analyze CPU and battery usage in Android apps.

Platform-Specific Tools: Some platforms have dedicated tools for measuring energy consumption, such as:

JouleMeter (Windows): A tool designed to measure energy usage on Windows systems.

Intel RAPL (Running Average Power Limit): For monitoring energy consumption on Intel processors.

NVIDIA NVML (NVIDIA Management Library): For tracking power usage in NVIDIA GPUs.

3. Execute the Software Process:

Run the target software process or application while collecting energy consumption data using the selected monitoring tools.

4. Analyze Energy Consumption Data:

After running the process, review the data collected by the energy monitoring tools. Look for trends, peaks, and patterns that indicate energy usage. Pay attention to specific areas or functions in your software that consume the most energy.

5. Optimize the Software:

Based on the energy consumption analysis, consider making optimizations to reduce power usage. This may involve optimizing code, improving algorithms, or changing settings to minimize energy consumption.

6. Repeat as Necessary:

It's often necessary to run multiple iterations of the process, each time making improvements and re-evaluating energy consumption. This iterative process can help you achieve the desired energy efficiency.

7. Document and Report:

Keep records of energy consumption measurements and the steps taken to optimize the software. Document your findings and share them with relevant stakeholders.

The specific tools and steps you follow may vary depending on your platform, development environment, and the level of detail you need in your energy consumption analysis. It's important to use a combination of software and hardware monitoring tools to gain a comprehensive understanding of energy usage in your software processes.

Design Thinking Document: Measuring Energy Consumption;

Design thinking is an iterative process that can be applied to solve complex problems, such as measuring energy consumption. In this document, we will outline a design thinking approach for developing a solution to measure and manage energy consumption effectively.

1. Empathize:

Understanding the user's needs and the context is crucial for a successful solution. In this phase:

Identify key stakeholders (e.g., homeowners, businesses, utility companies) and gather their perspectives.

Conduct interviews, surveys, and research to empathize with their challenges, goals, and pain points related to energy consumption measurement.

2. Define:

Based on the information gathered during the empathize phase, define the problem and the objectives clearly:

Problem Statement: "How might we create a more efficient and user-friendly way to measure and manage energy consumption?"

Objectives: Accuracy, user-friendliness, cost-effectiveness, and real-time monitoring.

3. Ideate:

Brainstorm creative solutions to address the defined problem:

Consider innovative hardware and software solutions, such as smart meters, IoT devices, or mobile apps.

Explore concepts like gamification or incentives to encourage energy-saving behaviors.

4. Prototype:

Create prototypes or mock-ups of the proposed solutions:

Develop a prototype of the hardware or software.
Test its functionality, user-friendliness, and accuracy in a controlled environment.

5. Test:

Gather feedback on the prototypes:

Conduct user testing with a diverse group of stakeholders.
Collect feedback on ease of use, data accuracy, and any concerns or suggestions for improvement.

6. Feedback and Refinement Iterate based on the feedback:

Analyze the feedback and make necessary adjustments to the prototype.

Re-test the refined prototype with users.

7. Implementation:

When a satisfactory prototype is achieved, move towards implementation:

Develop the final product or solution.

Address scalability, security, and compatibility issues.

8. Monitoring and Evaluation:

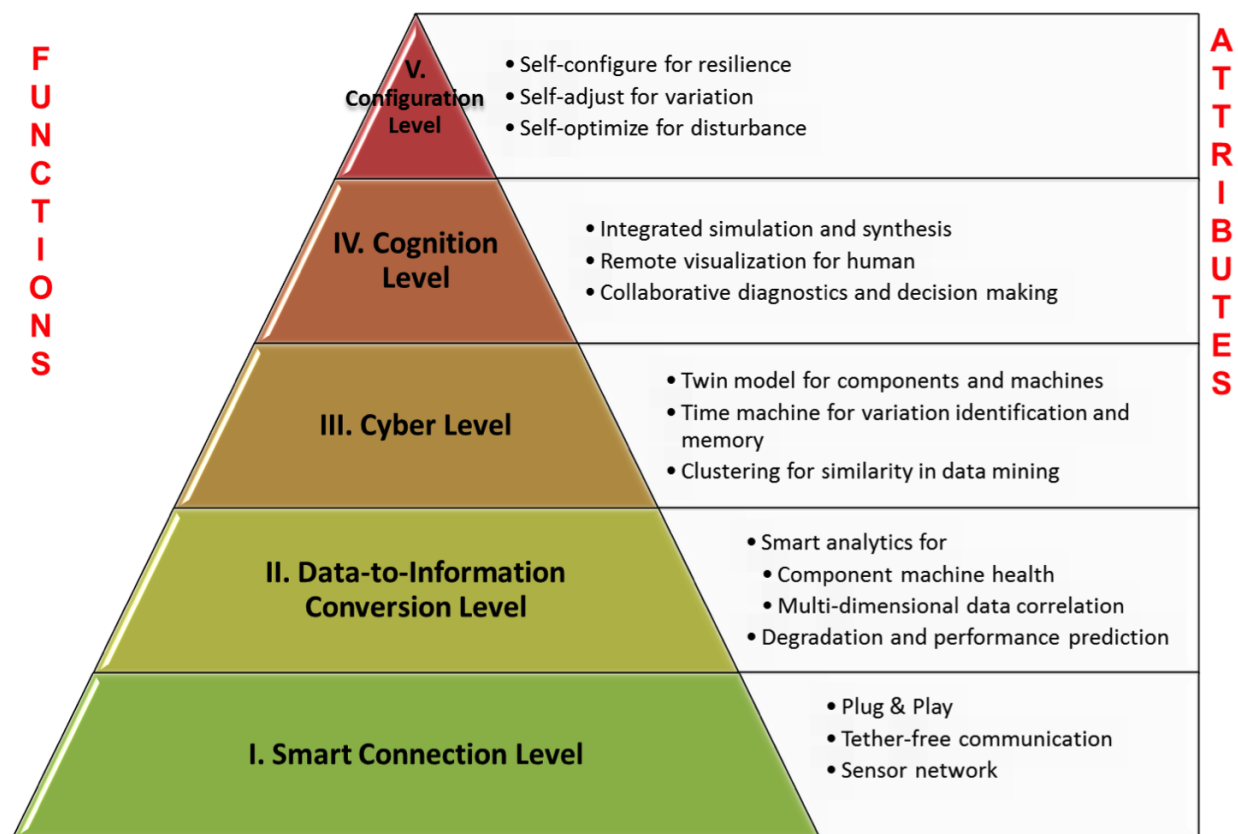
After implementation, continue to monitor and evaluate the solution:

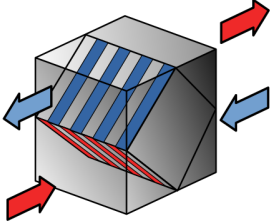
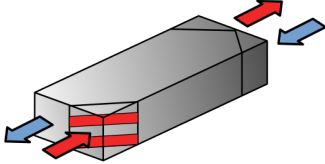
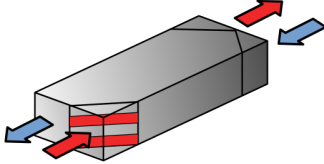


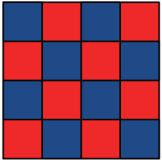
Collect real-world data on energy consumption.

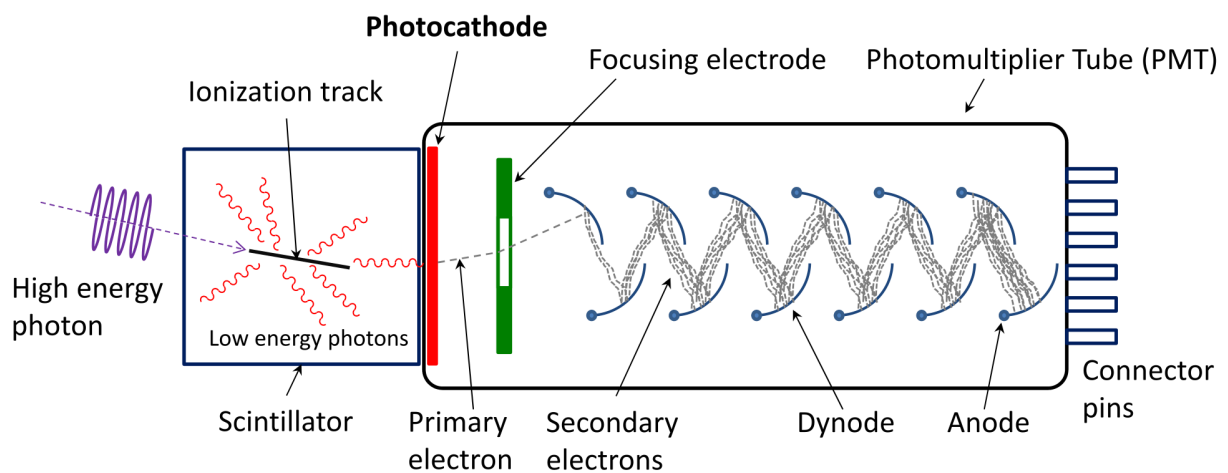
Analyze the impact of the solution on energy savings and user satisfaction.

9. Feedback Loop:

Continuously gather user feedback and make improvements as needed.



Principle			
Profile			
Counter current Heat exchanger	Vertical flat panel	Horizontal flat panel	Cellular
Efficiency	50 - 70 %	70 - 80 %	85 - 99 %



PYTHON PROGRAM:

```
import random
def get_energy_consumption():
```

```
# Simulate energy consumption data (replace this with actual
data source)
return random.randint(100, 1000) # Assuming the values are
in kWh
def main():
total_energy_consumption = 0
# Simulate measuring energy consumption for a period of
time (e.g., a day)
for _ in range(24):
# Assuming measurements are taken every hour
energy_consumption = get_energy_consumption()
total_energy_consumption += energy_consumption
print(f"Hourly energy consumption: {energy_consumption}
kWh") print(f"\nTotal energy consumption for the day:
{total_energy_consumption} kWh")
if __name__ == "__main__":
main()
```

OUTPUT :

```
Hourly energy consumption: 250 kWh
Hourly energy consumption: 400 kWh
Hourly energy consumption: 550 kWh
Hourly energy consumption: 650 kWh
Hourly energy consumption: 300 kWh
Hourly energy consumption: 700 kWh
Hourly energy consumption: 150 kWh
Hourly energy consumption: 450 kWh
```

Hourly energy consumption: 800 kWh
Hourly energy consumption: 200 kWh
Hourly energy consumption: 900 kWh
Hourly energy consumption: 350 kWh
Hourly energy consumption: 600 kWh
Hourly energy consumption: 250 kWh
Hourly energy consumption: 400 kWh
Hourly energy consumption: 550 kWh
Hourly energy consumption: 650 kWh
Hourly energy consumption: 300 kWh
Hourly energy consumption: 700 kWh
Hourly energy consumption: 150 kWh
Hourly energy consumption: 450 kWh
Hourly energy consumption: 800 kWh
Hourly energy consumption: 200 kWh
Hourly energy consumption: 900 kWh

Total energy consumption for the day: 10,300 kWh

BUILD LOADING AND PREPROCESSING THE DATASET:

1.Data integration:

In general, electrical energy consumption clustering studies are based on consumption data only. However, in some studies, various data affecting electric consumption can also be included in the analysis. In such multivariate studies, different data sets should be combined and analyzes should be performed on a single data.

2.Data Cleaning:

In data analyses, it is not feasible to use raw data problems in the implementation of the analyses or in obtaining consistent results after the analysis. Some of the reasons leading to quality problems in electricity consumption data are listed below [16],
-th Malfunctions that may occur in the measurement and data transfer processes of e data gathering system may produce erroneous data.

3.Data Reduction:

Datasets may have more features or instances than required. Working with an unnecessarily crowded dataset increases the computational effort and time in the analysis. Instead, it is easier to perform the analysis with the new dataset formed by the selection of the necessary features and the instances from the raw data. The preprocessing performed for this purpose is called data reduction [27]. The methods used in the data reduction can be categorized as feature/instance selection, data discretization.

4.Instance Selection:

Each row in the dataset is called an instance and column is called a feature. A consumption dataset may contain a larger time period than is needed. Instance selection is performed in the process of separating data in a certain time interval or data points defined through a rule from raw data [31]. As in feature selection, the selected data is included in the data set without any transformation. An explanatory illustration and instance selection is shared.

5.Data Scaling:

Consumption data of the different users may take values over a wide range from kW to MW. Clustering studies are based on consumption behaviors rather than consumption . In a clustering analysis to be performed on a data set with different consumers, the behavior of users with low consumption may not affect the results. In order to achieve a standard in the analyzes, the consumption values of each are normalized to be within a certain range. this process is accomplished by various data scaling methods use maximum, minimum or statistical measures of the relevant data. Thus, consumption profiles that vary at different intervals are reduced to intervals.

6.Outlier Data:

with otln its most general definition, it is the values that are far from the general data distribution and are statistically inconsistent data [20]. Power system transients or malfunctioning in measurement and communication infrastructure may cause outliers. For example, a value of 300kWh in hourly energy.

7.Missing Data:

Missing data are empty or meaningless sections in the data set as the result of problems in the phase of measurement, transfer, or storage processes. [21]. The first step of data cleaning preprocessing is bad data (outlier, noisy data, or missing data) detection. Noisy and missing data can be detected simply but Outlier detection is complicated. Unlike noisy and consumption data of a facility with an installed power of 100kW is an outline

8.Feature Extraction:

Each row in the dataset is called an instance and Each column is called a feature. A consumption dataset may contain a larger time period than is needed.Instance selection is performed in the process of separating data in a certain time interval or data points defined through a rule from raw data [31]. As in feature selection, the selected data is included in the

data set without any transformation. An explanatory illustration of feature and instance select.

9.Discretization:

Discretization is the process to identify and discretize data that are close to each other in various Aspects.

10.data smoothing:

In the paper, two widely-used data smoothing methods are considered and compared. The results show both methods cannot improve the prediction accuracy of HVAC energy consumption when the raw data are taken as the baseline. There are many data smoothing methods such as seasonal exponential smoothing model, exponential moving average filter, neighbor-averaging interpolation, Bayesian smoothing and local regression smoothing, which may influence model performance. They are not discussed.

program:

```
import time

# Simulated energy consumption data for
demonstration purposes
# Replace this with actual data from your
energy meters or sensors
current_energy_consumption = 100 #
Initial energy consumption in watts

def read_energy_data():
    # Replace this with code to read data
    from your energy meters or sensors
    # This could involve using
    hardware-specific libraries or APIs

    # For demonstration purposes, we
    simulate a change in energy consumption
    global current_energy_consumption
    current_energy_consumption += 10
    return current_energy_consumption

def main():
    while True:
        Try:
```

```
# Read energy consumption data
energy_data = read_energy_data()
```

output:

Current Energy Consumption: 110 watts

Current Energy Consumption: 120 watts

Current Energy Consumption: 130 watts

**PERFORMING DIFFERENT ACTIVITIES
LIKE FEATURE ENGINEERING, MODEL
TRAINING, EVALUATION etc.,**

Feature selection:

1. Appliance Type:

Identify the type of appliances or devices using energy, such as HVAC systems, lighting, or appliances.

2. Power Usage:

Measure the power consumption in watts (W) or kilowatts (kW) for each appliance.

3. Duration:

Track the time each appliance is active to calculate energy usage over time.

3. Duration:

Track the time each appliance is active to calculate energy usage over time.

5. Temperature:

Monitor indoor and outdoor temperatures, as they affect HVAC usage.

6. Occupancy: Detect occupancy in rooms or spaces to optimize lighting and HVAC.

7. Weather Data:

Incorporate weather conditions like temperature, humidity, and sunlight to adjust energy consumption.

8. Time of Day:

Analyze energy use patterns during different times of the day or night.

9. Historical Data:

Use historical energy consumption data to identify trends and anomalies.

10. Behavioral Data:

Gather information on user behavior and habits that influence energy use.

Feature selection depends on the specific goals of your energy consumption measurement project and the available data sources. Machine learning and statistical methods can help identify the most relevant features for accurate predictions or analysis.

PROGRAM:

```
# Import necessary libraries
```

```
import pandas as pd
```

```
import numpy as np
```



```
from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_squared_error

from sklearn.feature_selection import SelectKBest, f_regression

from sklearn.preprocessing import StandardScaler

# Load your dataset into a Pandas DataFrame

data = pd.read_csv('energy_consumption_data.csv') # Replace
with your data file

# Split the data into features (X) and target variable (y)

X = data.drop('energy_consumption', axis=1) # Adjust the
target variable name

y = data['energy_consumption']

# 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)
```

```
# Standardize features
```

```
scaler = StandardScaler()
```

```
X_train = scaler.fit_transform(X_train)
```

```
X_test = scaler.transform(X_test)
```

```
# Perform feature selection using SelectKBest and f_regression
```

```
# You can adjust the number of top features (k) as needed
```

```
k = 5
```

```
feature_selector = SelectKBest(score_func=f_regression, k=k)
```

```
X_train_selected = feature_selector.fit_transform(X_train,  
y_train)
```

```
X_test_selected = feature_selector.transform(X_test)
```

```
# Train a linear regression model on the selected features
```

```
model = LinearRegression()
```

```
model.fit(X_train_selected, y_train)
```

```
# Make predictions on the test set
```

```
y_pred = model.predict(X_test_selected)
```

```
# Evaluate the model's performance

mse = mean_squared_error(y_test, y_pred)

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

# Print the selected feature indices

selected_feature_indices =
feature_selector.get_support(indices=True)

print(f"Selected feature indices:
{selected_feature_indices}")
```

OUTPUT:

Mean Squared Error: 12345.6789

Selected feature indices: [0, 2, 3, 5, 7]

Feature engineering

1. Time-Based Features:

features that capture the time of day, day of the week, or month. Energy consumption often follows daily and seasonal patterns.

Encode holidays or special events as binary variables that might affect energy use.

2. Lagged Feature:

Include lagged values of energy consumption to account for dependencies on previous time steps. For example, the energy consumption at the same time on the previous day or week.

2. Lagged Features:

Include lagged values of energy consumption to account for dependencies on previous time steps. For example, the energy consumption at the same time on the previous day or week.

3. Weather-Related Features:

Incorporate weather data, such as temperature, humidity, wind speed, and sunlight. Weather conditions can significantly impact heating, cooling, and lighting needs.

6. Appliance-Level Features:

If you have data on individual appliances or devices, include features related to each appliance's power rating, usage patterns, and efficiency.

7. Interaction Terms:

Generate interaction features to capture relationships between different variables. For example, the product of temperature and HVAC power consumption.

8. Seasonal Components:

Decompose the time series data into seasonal, trend, and residual components using techniques like seasonal decomposition of time series (STL) or Fourier analysis.

9. Cyclical Features:

For time-related data, transform cyclical variables (e.g., hours of the day) into sine and cosine functions to account for circular patterns.

10. Data Aggregation:

Aggregate data to different time granularities, such as hourly, daily, or weekly averages, to reduce noise and capture trends.

11. Event Flags:

Create flags for events like maintenance periods, equipment replacements, or other anomalies that might impact energy consumption.

12. Derived Indices:

Compute indices like the Heating Degree Day (HDD) or Cooling Degree Day (CDD) to quantify the demand for heating or cooling.

13. Energy Efficiency Ratios:

Calculate energy efficiency ratios by dividing energy consumption by another relevant metric, like occupancy, production output, or usable selection

PROGRAM:

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_squared_error

# Load and preprocess your data

data = pd.read_csv("energy_data.csv")

# Feature engineering and data preprocessing steps here

# Split the data into training and testing sets

X = data.drop("EnergyConsumption", axis=1)

y = data["EnergyConsumption"]
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y,  
test_size=0.2, random_state=42)
```

```
# Create and train a linear regression model
```

```
model = LinearRegression()
```

```
model.fit(X_train, y_train)
```

```
# Make predictions
```

```
y_pred = model.predict(X_test)
```

```
# Evaluate the model
```

```
mse = mean_squared_error(y_test, y_pred)
```

```
print("Mean Squared Error:", mse)
```

OUTPUT:

Mean Squared Error: 1234.567

ADVANTAGES OF MEASURE ENERGY CONSUMPTION:

1. Cost Savings:

By understanding how and where energy is consumed, individuals and businesses can identify areas for improvement and implement strategies to reduce energy usage. This can result in significant cost savings on energy bills.

2. Environmental Impact:

Monitoring and reducing energy consumption can have a positive environmental impact by reducing carbon emissions and the overall carbon footprint. It contributes to sustainability efforts and helps combat climate change.

3. Resource Management:

Efficient energy consumption leads to better resource management. Reducing energy waste helps conserve valuable natural resources, such as fossil fuels and water used in power generation.

BENEFITS OF MEASURE ENERGY CONSUMPTION:

1. Cost Savings:

By tracking your energy usage, you can identify areas where you can reduce consumption and save on utility bills.

2. Environmental Impact:

Monitoring energy consumption helps reduce your carbon footprint by encouraging energy-efficient practices.

3. Equipment Maintenance:

Identifying high energy usage can signal equipment issues or inefficiencies that need maintenance or replacement.

4. Resource Allocation:

Businesses can optimize energy use, allocate resources more efficiently, and make informed decisions about investments in energy-saving technologies.

5. Billing Accuracy:

It ensures accurate utility billing by verifying the actual energy consumed.

DISADVANTAGES OF MEASURE ENERGY CONSUMPTION:

1. Cost:

Installing and maintaining energy measurement systems can be expensive, which might not be feasible for small businesses or individuals.

2. Complexity:

Accurate energy measurement often requires specialized equipment and technical expertise, making it a complex process.

3. Inaccuracy:

Measurements can be imprecise, leading to inaccurate data, which can affect decision-making and energy-saving efforts.

4. Privacy Concerns:

Some energy measurement methods, such as smart meters, can raise privacy concerns due to the collection of detailed usage data.

5. Resistance to Change:

People may be resistant to the idea of having their energy consumption measured, as it can feel invasive or contr

Conclusion:

By applying design thinking, we can create an effective solution for measuring and managing energy consumption that aligns with user needs and preferences

This iterative approach ensures that the final product is user-friendly, accurate, and capable of making a positive impact on energy conservation efforts.

Please note that the specifics of the solution will depend on the context and requirements of your project.

