INTEL-UNNATI JOURNAL
Number 2 (2024), pp. 1–22

Code:IUCP-2024

TEAM SPOTLIGHT

Power Manager Telemetry

Ashly Biju, Midhun Mathew, Sharan Susan Aby, Roshan R John , and Blesson Biji

Saintgits Group of Institutions, Kottayam, Kerala

Abstract:
The rapid expansion of 5G networks and edge computing has led to a significant increase in distributed device deployment, resulting in increased power consumption across various industries. This project addresses the critical need for efficient power management in the context of rising electricity costs and government initiatives for net zero consumption. The primary objective is to develop a comprehensive system for measuring and analyzing power utilization in computer systems, with a focus on CPU, memory, network interface card (NIC), and thermal design power (TDP) components.This research identifies and leverages open-source tools for accurate power measurement, documents available system knobs for power telemetry, and implements a solution to collect and analyze power consumption data. The project utilizes containerization technology like docker to simulate various system utilization scenarios, allowing for a detailed examination of the relationship between workload and power draw.The findings of this study contribute to a deeper understanding of power utilization in modern computing systems and offer potential strategies for optimization. This work aligns with broader sustainability goals in the tech industry and provides a foundation for future research in energy-efficient computing.

Keywords: Telemetry,Edge Computing,Net Zero Power consumption,Network Interface Card(NIC),Thermal Design Power(TDP)

1 Introduction

The rapid expansion of 5G networks and edge computing has led to a significant increase in power consumption across the IT sector [3] .This increase in energy usage presents both economic and environmental challenges and as organizations struggle with increasing electricity costs, governments worldwide are simultaneously pushing for ambitious net-zero power consumption initiatives, adding regulatory pressure to an already complex landscape.In response to these pressing concerns, the Power Manager Telemetry project emerges as a critical endeavor to address the urgent need for efficient power management in modern computer systems.This comprehensive project sets out to develop an advanced system for measuring and analyzing power utilization, with a particular focus on key components such as CPU, memory, network interface card (NIC), and thermal design power (TDP). The project's multifaceted approach encompasses several crucial objectives:

1. Implement open-source tools for precise power measurement.

2 .Identify system knobs for measuring power consumption in various components (CPU, memory, NIC, TDP).

3. Develop a methodology for collecting and analyzing power telemetry data.

4. Create a solution to measure system power utilization based on specified parameters.

By pursuing these ambitious goals, this project aims to make a significant contribution to energy efficiency efforts within the IT sector, aligning closely with broader sustainability initiatives that are becoming increasingly vital in the tech industry. [6]. The project's outcomes are expected to equip organizations with the tools and knowledge necessary to better manage their power consumption in the rapidly evolving landscape of 5G and edge computing.The insights and solutions derived from this project have the potential to influence industry practices, inform policy decisions, and pave the way for further innovations in energy-efficient computing, ultimately contributing to a more sustainable and resilient IT infrastructure in the face of growing environmental concerns and energy demands.

2 TOOLS IDENTIFIED

In the power management telemetry project, we utilized various tools on both Windows and Linux platforms to monitor, analyze and optimize power cosumption. Below are the various tools used.

2.1 Windows

2.1.1 Intel One Gadget

Intel One Gadget is a comprehensive power estimation and analysis tool developed by Intel. It allows users to evaluate the power consumption of various components in a system, providing detailed insights into energy usage and efficiency. With capabilities to model and simulate power scenarios, Intel One Gadget is instrumental in optimizing power management strategies and achieving energy-efficient designs.

2.1.2 Open Hardware Monitor

OpenHardware Monitor is a free and open-source software for monitoring temperature sensors, fan speeds, voltages, and power consumption. In this project, it was used to collect power consumption data, enabling detailed analysis and optimization of energy usage.

2.1.3 HWiNFO

HWiNFO is a powerful hardware monitoring and diagnostic tool that provides comprehensive details about system components, including real-time monitoring of CPU, GPU, memory, and power consumption. It was used in this project to gather detailed power consumption data and system performance metrics.

2.1.4 CPU-Z

CPU-Z is a sofware system profiling and monitoring tool for Windows that gathers information on the central processing unit (CPU), RAM, motherboard, and other hardware components. In this project, CPU-Z was used to analyze power consumption and performance metrics of the CPU.

2.2 LINUX

2.2.1 Cpupower

CPU Power is a Linux tool used for managing and monitoring CPU frequency scaling and power consumption. It provides detailed information and control over the CPU's power states and performance levels. In this project, CPU Power was utilized to monitor and optimize CPU power consumption on Linux systems.

2.2.2 PowerStat

PowerStat is a Linux tool used to measure power consumption and energy usage of a system. It provides detailed reports on power usage, which are essential for analyzing and optimizing power management strategies. In this project, PowerStat was utilized to monitor and analyze the power consumption of the system, aiding in the identification of areas for energy efficiency improvements.

2.2.3 PowerTop

PowerTOP is a Linux tool designed to diagnose issues with power consumption and provide suggestions for optimizing power usage. It monitors system activity and identifies processes and components that consume the most power.

2.2.4 Psutil

psutil is a cross-platform library for retrieving information on system utilization (CPU, memory, disks, network, sensors) and system uptime. It is particularly useful for monitoring and profiling system performance.

2.2.5 Turbostat

Turbostat is a Linux tool used for monitoring CPU performance and power consumption. It provides detailed statistics on processor frequency, power consumption, and temperature.

2.2.6 VMstat

vmstat is a command-line utility available on Unix-like operating systems, including Linux. It provides reports on system-wide statistics for processes, memory, paging, block I/O, traps, and CPU activity. VMstat was used to monitor and analyze system performance metrics, including CPU utilization and memory usage, to optimize power consumption and system efficiency.

2.2.7 Some command line utility tools in Linux

Measuring CPU, NIC, Memory, and TDP power consumption in Linux can be achieved using various command-line utilities. Here are some commonly used tools for each:

ethtool : ‘ethtool‘ is used to query and control network driver and hardware settings, including power management settings for NICs.

nload : ‘nload‘ provides a visual representation of network traffic and bandwidth usage.]

free : -‘free‘ displays the amount of free and used memory in the system.

smem : ‘smem‘ is a tool that provides a more accurate view of memory usage by reporting PSS (Proportional Set Size).

s-tui : ‘s-tui‘ is a terminal UI for monitoring CPU temperature, frequency, power, and utilization in real-time.

3 Methodology

Research on Open-Source Power Measurement Tools: We investigated several open-source tools for power measurement,for the windows and linux operating systems. These tools were compared based on accuracy, ease of use, and compatibility with our target systems.

System Power Measurement Knobs: We identified key power measurement knobs:
- CPU: Frequency scaling, C-states, P-states
- Memory: DRAM power states
- NIC: Power management states, packet coalescing
- TDP: Power capping interfaces

Some other identified knobs include:

- CPU Package Power [W]
- IA Cores Power [W]
- GT Cores Power [W]
- Total System Power [W]
- System Agent Power [W]
- Rest-of-Chip Power [W]
- PCH Power [W]
- Core Usage (avg) [

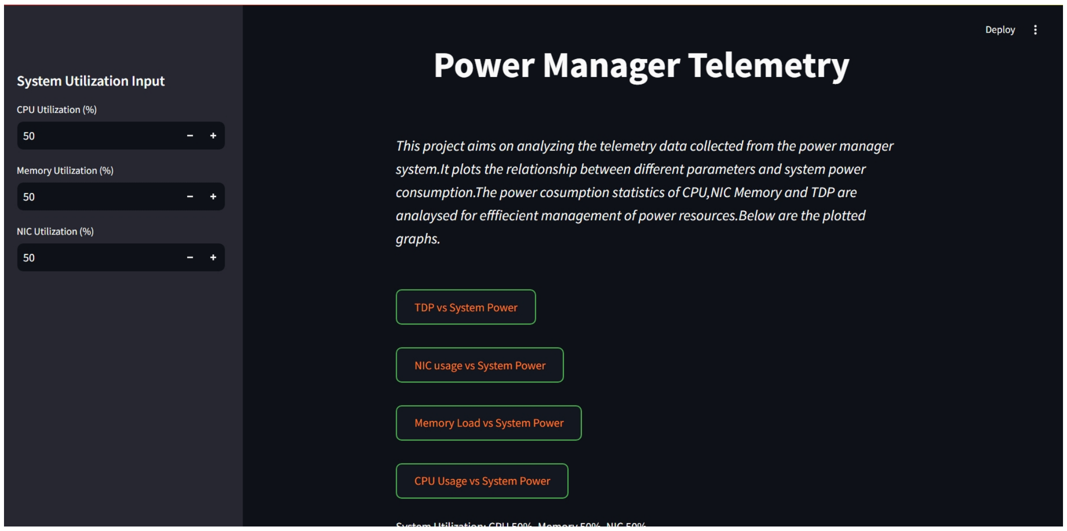
- Core Temperatures (avg) [C]
- Physical Memory Load [
- Memory Clock [MHz]
- Current cTDP Level []
- Package C2 Residency [
- Voltage [V]
- Remaining Capacity [Wh]

Data Collection Methodology :A python was run in the windows system which collects power telemetry data for various system knobs.This script automates the process of opening HWINFO and OpenHardwareMonitor applications at each 5% battery reduction, from 100% to 5%. It waits until the battery level reaches the specified percentage, then runs both applications, allows them to log data for a specified duration, and saves the logs in separate folders named 'HWiNFO Logs' and 'OpenHardwareMonitor Logs'. The log files are named according to the current battery percentage. This ensures consistent logging of system data at defined battery levels and was saved in a csv file.In linux system the data was colleted using linux tools like powerstat,turbostat,vmstat.We implemented real-time data visualization using matplotlib from python to monitor power consumption trends.

System Utilization Simulation: We used Docker containers to create isolated environments for simulating various system loads. Different utilization levels were achieved through stress tests by running CPU-intensive (e.g., prime number calculation), memory-intensive (e.g., large array operations), and network-intensive (e.g., iperf3) workloads. Traffic generation was managed using custom scripts that adjust the intensity of workloads to match desired utilization percentages.

This approach allowed us to systematically measure and analyze power consumption under controlled conditions, providing insights into the relationship between system utilization and power draw.

Creation of GUI : A GUI was created using streamlit framework for the project to showcase the power consumption trends of various system knobs to the system power utilisation.The plots for CPU,NIC,TDP and Memory Vs system power cosumption is shown.An input and output mechanism providing results of power consumed in watts different for utilisation percentages of system knobs was also created.



Power Manager Telemetry

This project aims on analyzing the telemetry data collected from the power manager system.It plots the relationship between different parameters and system power consumption.The power cosumption statistics of CPU,NIC Memory and TDP are analysed for efffiecient management of power resources.Below are the plotted graphs.

System Utilization: CPU 50%, Memory 50%, NIC 50%

Figure 1: Streamlit UI

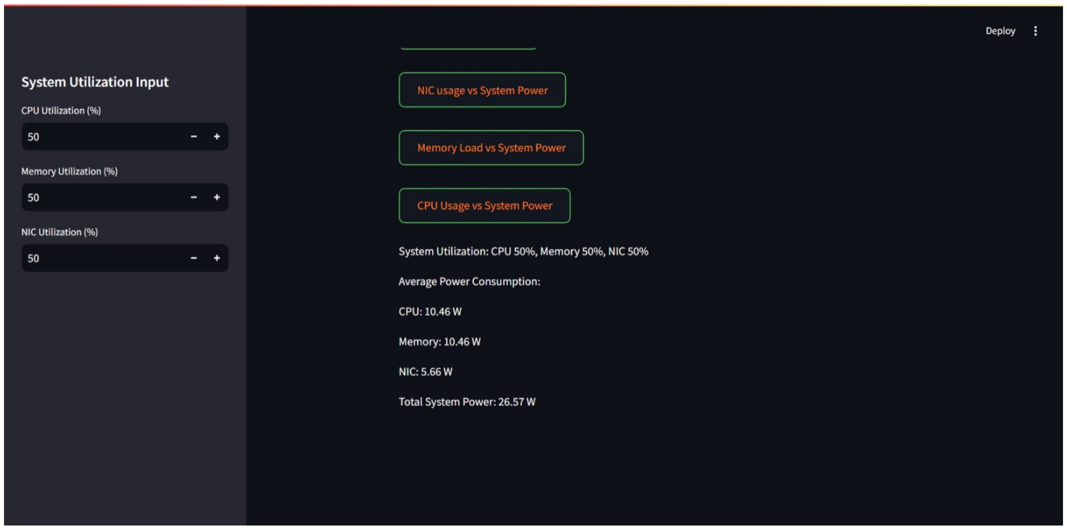
4 Implementation

Description of the final solution: Our final solution is a Python-based application that measures power consumption based on system utilization. It integrates the data collection methods from our research phase with a user-friendly interface for specifying utilization levels. The solution uses a wide range of tools mentioned above for comprehensive system-wide power analysis.

Key components :
1. Power measurement module using specialised tools for different os
2. System utilization controller and isolated environment creation using Docker containers.
3. Data aggregation ,plotting of graphs of various system utilisation knobs to the power consumed .
4. User interface for input and result display

Input/output mechanism :
Input:
Users specify desired system utilization percentage via the streamlit user interface.
Output:
System power consumption data displayed in the user interface Summary report with average power consumption for each component (CPU, memory, NIC, Total System Power)
Graphical representation of power usage vs. utilization
Example output:
System Utilization: CPU 75%, Memory 60%, NIC 30%
Average Power Consumption:

CPU: 45.2 W
Memory: 12.8 W
NIC: 3.5 W
Total System Power: 127.2 W


Deploy
NIC usage vs System Power
Memory Load vs System Power
CPU Usage vs System Power

System Utilization Input

CPU Utilization (%)
50

Memory Utilization (%)
50

NIC Utilization (%)
50

System Utilization: CPU 50%, Memory 50%, NIC 50%

Average Power Consumption:

CPU: 10.46 W

Memory: 10.46 W

NIC: 5.66 W

Total System Power: 26.57 W

Figure 2: Input and Output for various System utilisation

5 Results & Discussion

We identified various system knobs and plotted graphs for the same with respect to the system power utilisation.

5.1 CPU Usage vs System Power Consumption

Insights from the Plot :

- **Linear Relationship:** The plot shows a clear linear relationship between average CPU usage (%) and average system power consumption (W).
- **Minimal Variation at Low Usage:** At low CPU usage percentages (0-5%), the average system power consumption remains relatively stable, slightly above 10W.
- **Significant Increase with Higher Usage:** As CPU usage increases beyond 5%, there is a noticeable and steady increase in system power consumption, reaching around 20W at 30% CPU usage.
- **High Correlation:** The plot suggests a high correlation between CPU usage and power consumption, implying that as the CPU usage increases, the system power consumption also increases proportionally.

Key Points :

- **Linear Trend:** Indicates that system power consumption can be predicted based on CPU usage.
- **Stability at Low Usage:** Suggests efficient power management at lower CPU utilizations.
- **Significant Rise with High Usage:** Highlights the impact of higher CPU usage on overall system power consumption.

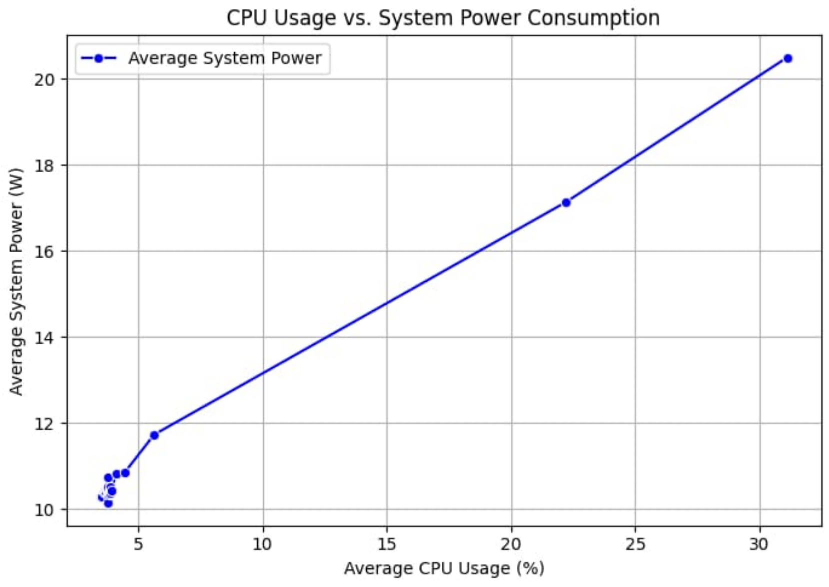


Figure 3: CPU Usage vs. System Power Consumption

5.2 NIC Usage vs System Power Consumption

Insights from the Plots :

- **Download Rate vs. System Power:**
  - Positive linear correlation between download rate and system power consumption.
  - Download rates range from approximately 610 KB/s to 825 KB/s.
  - Power consumption increases from approximately 10W to 20W.
  - Data points are clustered at lower download rates.
- **Upload Rate vs. System Power:**
  - Positive linear correlation between upload rate and system power consumption.
  - Upload rates range from approximately 50 KB/s to 100 KB/s.
  - Power consumption increases from approximately 10W to 20W.
  - Fewer data points which are more spread out compared to download rate data.

Key Points :

- Both upload and download activities increase power consumption.
- Download rates are significantly higher than upload rates.
- Similar power range (10-20W) for both activities.
- Download shows more clustered low-rate, low-power usage.
- Linear relationships suggest predictable power scaling with network activity.

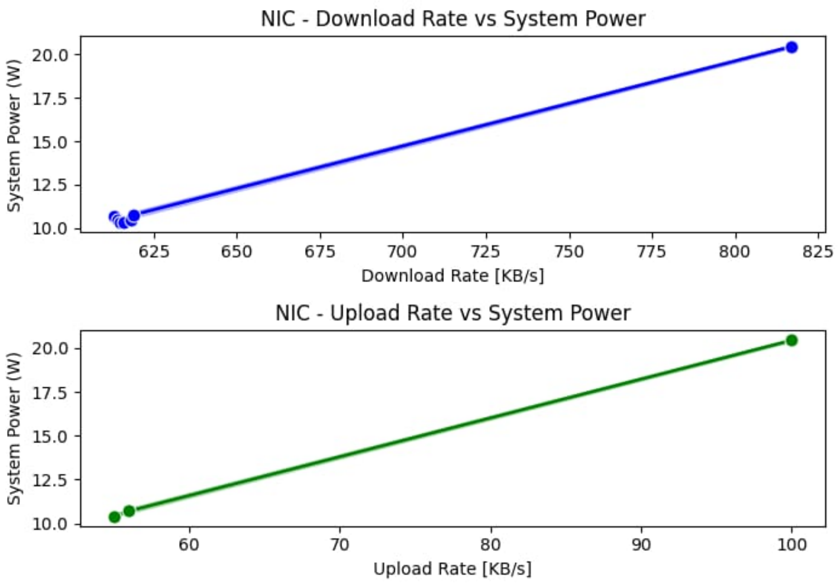


Figure 4: Network Activity vs. System Power Consumption

5.3 Memory Usage vs System Power Consumption

Insights from the Plot :

- **Physical Memory Usage vs. System Power:**
  - Non-linear relationship with sharp spikes.
  - Memory usage ranges from approximately 61% to 67%.
  - Power consumption varies between approximately 10W and 21W.
  - Most data points cluster around 10-11W.

Key Points :

- Power consumption generally stable across memory usage.
- Three distinct spikes to approximately 21W at different usage levels.
- Highest spike occurs at lowest memory usage (approximately 61%).
- Sudden power increases suggest specific system events.
- Other factors likely contribute to observed power spikes.

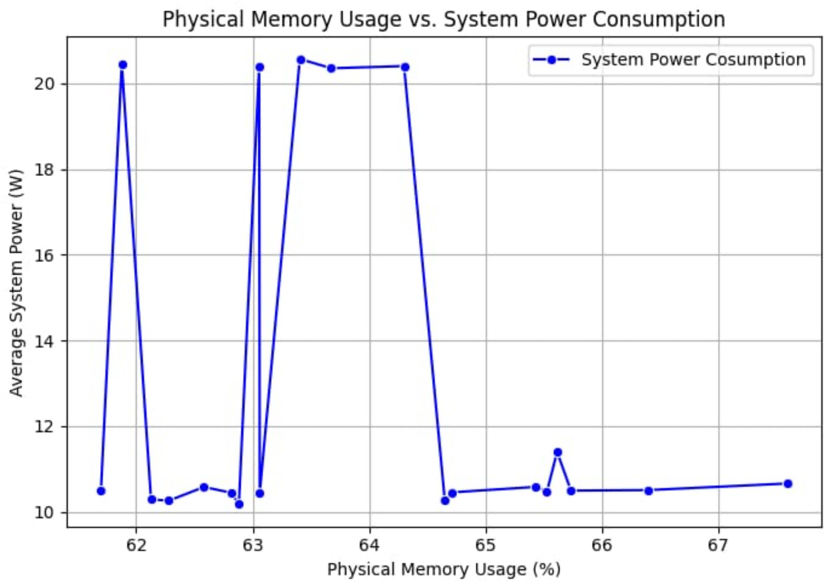


Figure 5: Physical Memory Usage vs. System Power Consumption

5.4 TDP Usage vs System Power Consumption

Insights from the Plots :

- **CPU Package Temperature vs CPU Package Power:**
  - Non-linear positive correlation.
  - Temperature range: approximately 48°C to 92°C.
  - Power range: approximately 6W to 15W.
  - Sharp increase in power above 75°C.
  - Fluctuations in mid-range temperatures.
- **Core Temperature vs Total System Power:**
  - Generally positive correlation with fluctuations.
  - Temperature range: approximately 47°C to 82°C.
  - Power range: approximately 10W to 20W.
  - Sharp spikes in power at approximately 72°C and 76°C.
  - More stable power consumption at lower temperatures.

Key Points :

- Both CPU package and core temperatures correlate with increased power consumption.
- CPU package power shows steeper increase at high temperatures.
- Total system power has more pronounced spikes and fluctuations.
- Power consumption more stable at lower temperatures for both metrics.
- CPU package temperature reaches higher values than core temperature.
- Significant power increase occurs around 75-80°C in both plots.
- Data suggests thermal throttling or increased cooling efforts at high temperatures.

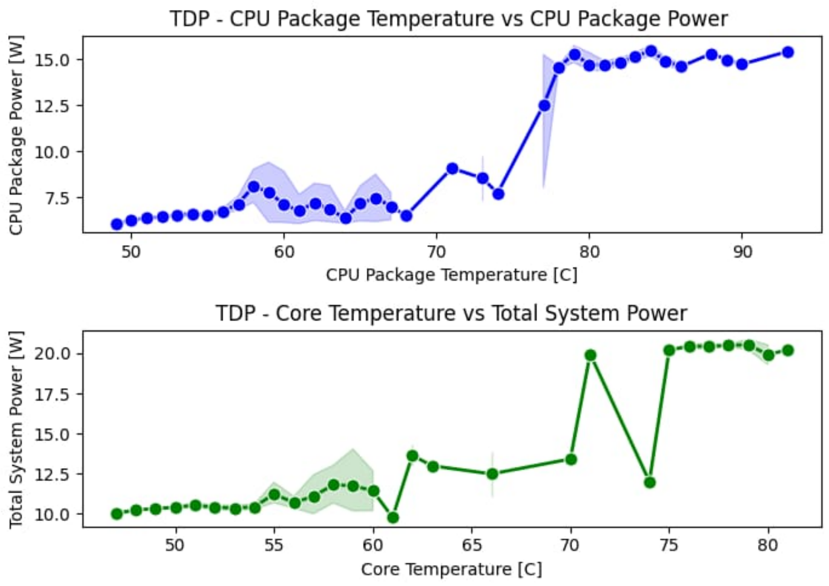


Figure 6: CPU Package Temperature vs CPU Package Power and Core Temperature vs Total System Power

1.

Table 1: Summary of various system utilisation to the power consumed

Sl No.	CPU Utilization	NIC Utilization	Memory Utilization	CPU (W)	NIC (W)	Memory (W)	Total System Power (W)
1.	50%	50%	50%	10.46W	10.46W	5.66W	26.57W
2.	75%	30%	50%	10.26W	10.46W	5.66W	26.37W
3.	100%	25%	50%	10.66W	20.38W	5.66W	36.70W
4.	10%	25%	60%	20.45W	20.38W	6.79W	47.63W
5.	25%	25%	25%	20.38W	20.38W	2.83W	43.60W
6.	40%	25%	35%	10.50W	20.38W	3.96W	34.84W

From Table 1, we get insights on the utilization of various system components (CPU, NIC, Memory) and their corresponding power consumption (in watts), along with the total system power.

CPU Utilization vs. CPU Power :

- The CPU power consumption is fairly consistent, with minor variations between 10.46W and 20.38W.
- At higher CPU utilization (100%), the CPU power increases slightly to 10.66W, and for other lower utilizations, it varies between 10.46W and 20.38W.

NIC Utilization vs. NIC Power :

- The NIC power consumption is also fairly consistent, around 10.46W to 20.38W across various utilization percentages.
- Even at the lowest NIC utilization (10%), the power consumption is 10.46W, indicating a minimal change in power consumption with varying utilization.

Memory Utilization vs. Memory Power :

- Memory power consumption varies more noticeably than CPU and NIC, ranging from 2.83W to 6.79W.
- Higher memory utilization (60%) correlates with higher power consumption (6.79W), while lower utilization (25%) correlates with lower power consumption (2.83W).

Total System Power :

- The total system power consumption varies significantly from 26.37W to 47.63W.
- Higher combined utilizations (Sl. No. 4) lead to higher total system power consumption (47.63W).
- Lower combined utilizations (Sl. No. 2) correspond to lower total system power consumption (26.37W).

General Trends :

- There is a direct correlation between utilization percentages and total system power consumption.
- Memory utilization seems to have a more pronounced effect on power consumption compared to CPU and NIC.

5.5 Key Observations

- **Stability of CPU and NIC Power Consumption:** CPU and NIC power consumption remains relatively stable despite varying utilization percentages, suggesting efficient power management.
- **Variable Memory Power Consumption:** Memory power consumption shows more variation with utilization changes, which impacts the total system power consumption.
- **Total System Power Dependency:** The total system power consumption is influenced more by memory utilization variations than by CPU and NIC utilization variations.

6 Conclusions

This project successfully developed a comprehensive system for measuring and analyzing power utilization in computer systems, addressing the challenges of increased energy consumption in 5G and edge computing environments. We achieved our objectives by identyfying open-source tools, documenting system knobs for power telemetry using some of those tools, and creating a system for measuring power usage based on system utilization.Our findings provide valuable insights into the relationship between workload and power consumption, contributing to the goal of net-zero power consumption in the IT sector. This work lays a foundation for future optimization strategies and more energy-efficient practices in computing and telecommunications.The tools and methodologies developed offer organizations a means to better understand and manage their power usage, potentially reducing operational costs and environmental impact. As technology continues to evolve,this research paves way for further studies in predictive power management and AI-driven optimization, supporting a more sustainable future in IT.

Acknowledgments

We would like to express our heartfelt gratitude and appreciation to Intel© Corporation for providing an opportunity to this project.First and foremost, we would like to extend our sincere thanks to our team mentor Pradeep C for his invaluable guidance and constant support throughout the project.We are deeply indebted to our college Saintgits College of Engineering and Technology for providing us with the necessary resources,and sessions on machine learning. We extend our gratitude to all the researchers, scholars, and experts in the field of power management and measurement, whose seminal work has paved the way for our project. We acknowledge the mentors, institutional heads, and industrial mentors for their invaluable guidance and support in completing this industrial training under Intel© -Unnati Programme whose expertise and encouragement have been instrumental in shaping our work.

[]

References

[1] BENINI, L., BOGLIOLO, A., AND DE MICHELI, G. A survey of design techniques for system-level dynamic power management. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems 8*, 3 (2000), 299–316. 10.1109/92.845896.

[2] BOLLINGER, T. Linux in practice: an overview of applications. *IEEE Software 16*, 1 (1999), 72–79. 10.1109/52.744572.

[3] BURNETT, R., BUTTS, M., AND STERLINA, P. Power system applications for phasor measurement units. *IEEE Computer Applications in Power 7*, 1 (1994), 8–13. 10.1109/67.251311.

[4] KEMPI, I., AHMED, N., HAMMER, A., OLABODE, O., UNNIKRISHNAN, V., KOSUNEN, M., AND RYYNÄNEN, J. A low-power hardware stack for continuous data streaming from telemetry implants. In *2018 IEEE Nordic Circuits and Systems Conference (NORCAS): NORCHIP and International Symposium of System-on-Chip (SoC)* (2018), pp. 1–6. 10.1109/NORCHIP.2018.8573522.

[5] RODRIGUEZ, M., STAHL, G., CORRADINI, L., AND MAKSIMOVIC, D. Smart dc power management system based on software-configurable power modules. *IEEE Transactions on Power Electronics 28*, 4 (2013), 1571–1586. 10.1109/TPEL.2012.2209681.

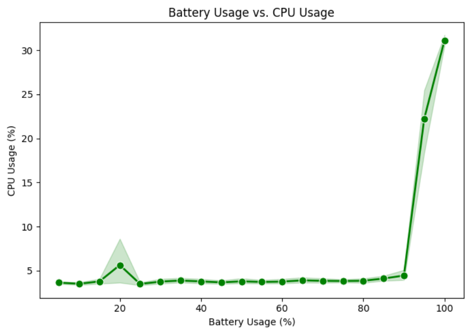
[6] ZHAO, J., YAO, L., XUE, R.-F., LI, P., JE, M., AND XU, Y. P. An integrated wireless power management and data telemetry ic for high-compliance-voltage electrical stimulation applications. *IEEE Transactions on Biomedical Circuits and Systems 10*, 1 (2016), 113–124. 10.1109/TBCAS.2015.2404038.

A Libraries Used

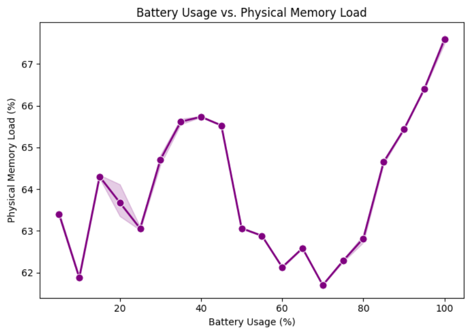
```
pandas
subprocess
matplotlib
csv
os
time
datetime
psutil
```

B System Power Utilisation and Battery Level

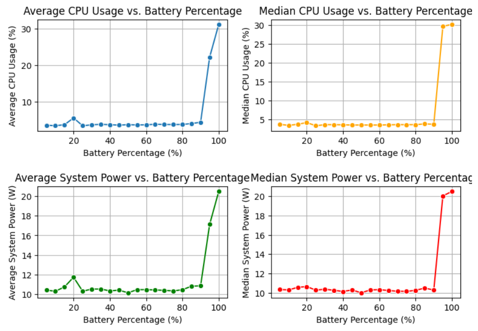
During the identification of various System knobs Vs System Power Consumption we also identified another factor that indirectly affects them called battery uage or battery level.Below are the plots showing some of our identifications



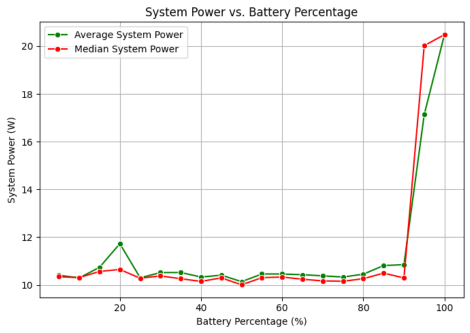
(a) Battery Usage Vs Cpu Usage



(b) Battery Usage Vs Physical Memory Load



(c) Cpu Usage Vs Battery Percentage and System Power Vs Battery Percentage



(d) System Power Vs Battery Percentage

Figure 7: Visualization of System knobs Vs Battey Level

C Main code sections for the solution

C.1 Data collection Script for Windows using tools Open Hardware and HWiNFO

```
def create_folders():
    if not os.path.exists("HWiNFO_Logs"):
        os.makedirs("HWiNFO_Logs")
    if not os.path.exists("OpenHardwareMonitor_Logs"):
        os.makedirs("OpenHardwareMonitor_Logs")
```

```
    def run_hwinfo():
    hwinfo_path = r'C:\Users\Midhun Mathew\Desktop\INTEL\HWiNFO_WINDOWS\HWiNFO64.
                                          exe'
    process = subprocess.Popen([hwinfo_path])
    return process
```

```
    def run_openhardwaremonitor():
    openhardwaremonitor_path = r'C:\Path\To\OpenHardwareMonitor.exe'
    process = subprocess.Popen([openhardwaremonitor_path])
    return process
```

```
    def save_logs(app_name, battery_percentage):
    source_path = r'C:\Path\To\Log\File.csv'  # Update with the actual path where
                                          logs are saved
    if app_name == "HWiNFO":
        destination_folder = "HWiNFO_Logs"
    else:
        destination_folder = "OpenHardwareMonitor_Logs"

    destination_path = os.path.join(destination_folder, f"{battery_percentage}%.
                                          csv")
    shutil.copyfile(source_path, destination_path)
```

```
   def main():
    create_folders()
    battery_levels = list(range(100, 0, -5))

    for level in battery_levels:
        # Wait until battery level reaches the desired percentage
        while psutil.sensors_battery().percent > level:
            time.sleep(60)  # Check every minute

        # Run HWiNFO and OpenHardwareMonitor
        hwinfo_process = run_hwinfo()
        openhardwaremonitor_process = run_openhardwaremonitor()

        # Give some time for the applications to start and log data
        time.sleep(300)  # Log for 5 minutes (adjust as needed)

        # Save the logs
        save_logs("HWiNFO", level)
        save_logs("OpenHardwareMonitor", level)

        # Terminate the processes
        hwinfo_process.terminate()
        openhardwaremonitor_process.terminate()

if __name__ == "__main__":
    main()
```

C.2 Data collection script for both os using psutil

```
def log_stats(duration=60):
    """Logs CPU, memory, power (if available), and NIC usage to a CSV file.

    Args:
        duration (int, optional): The duration (in seconds) to log data. Defaults
                                          to 60.
    """

    fields = [
        "timestamp",
        "cpu_usage",
        "memory_usage",
        "power_usage",
        "bytes_sent",
        "bytes_received",
    ]
```

```
    # Get the desktop path using platform-specific methods
    desktop_path = os.path.join(os.path.expanduser("~"), "Desktop")

    # Create the filename with timestamp for uniqueness
    filename = f"system_usage_{datetime.now().strftime('%Y-%m-%d_%H-%M-%S')}.csv"
    filepath = os.path.join(desktop_path, filename)

    with open(filepath, "w", newline="") as csvfile:
        writer = csv.writer(csvfile)
        writer.writerow(fields)

        start_time = time.time()
        while time.time() - start_time < duration:
            timestamp = datetime.now().strftime("%H:%M:%S")
            cpu_usage = psutil.cpu_percent()
            memory_usage = psutil.virtual_memory().percent
```

```
    # Get NIC usage (bytes sent and received)
            net_io_counters = psutil.net_io_counters(
                pernic=False
            )  # Get total NIC usage
            bytes_sent = net_io_counters.bytes_sent
            bytes_received = net_io_counters.bytes_recv

            writer.writerow(
                [
                    timestamp,
                    cpu_usage,
                    memory_usage,
                    power_usage,
                    bytes_sent,
                    bytes_received,
                ]
            )
            csvfile.flush()

    print(f"Logging finished. Data saved to {filepath}")
    return filepath
```

C.3 Plot from data collected using psutil

```
    def plot_data(filepath):
    data = pd.read_csv(filepath)
    timestamps = data["timestamp"]

    # Convert non-numeric columns to numeric
    data["power_usage"] = pd.to_numeric(data["power_usage"], errors="coerce")

    columns = [
        "cpu_usage",
        "memory_usage",
        "power_usage",
        "bytes_sent",
        "bytes_received",
    ]
    window_size = 5  # Window size for rolling average

    for column in columns:
        plt.figure(figsize=(10, 5))
        if column in ["bytes_sent", "bytes_received"]:
            # Scale NIC data for better visualization
            plt.plot(
                timestamps, data[column] / 1e6, label=column, alpha=0.5
            )  # Convert to MB
            plt.ylabel(f"{column} (MB)")
        else:
            # Smooth the values using a rolling average
            smoothed_values = (
                data[column].rolling(window=window_size, min_periods=1).mean()
            )
            plt.plot(
                timestamps, smoothed_values, label=f"{column} (smoothed)", alpha=0
                                                        .5
            )
            plt.ylabel(column)

        plt.xlabel("Time")
        plt.title(f"{column} over Time")
        plt.xticks(rotation=45)
        plt.legend()
        plt.tight_layout()
        plt.show()


if __name__ == "__main__":
    log_filepath = log_stats(duration=60)
    plot_data(log_filepath)
```

C.4 plot for tdp vs system power consumption

```
df = pd.read_csv(data_file)

# Set up the matplotlib figure
plt.figure(figsize=(7, 5))

# Plot Core Temperatures vs Total System Power
sns.lineplot(
    data=df,
    x="Core Temperatures (avg) [C]",  # Replace with your actual column for Core
                                             Temperatures
    y="Total System Power [W]",  # Replace with your actual column for System
                                             Power
    marker="o",
    color="blue",
)
plt.title("Core Temperatures vs System Power Consumption")
plt.xlabel("Core Temperatures [C]")
plt.ylabel("System Power Consumption [W]")

# Adjust layout
plt.tight_layout()

# Show plot
plt.show()
```

C.5 plot for NIC usage vs system power consumption

```
df = pd.read_csv(data_file)

# Set up the matplotlib figures
plt.figure(figsize=(7, 5))

# Plot for Download Rate vs System Power
plt.subplot(2, 1, 1)
sns.lineplot(
    data=df,
    # x="Current DL rate [KB/s]",  # Replace with your actual column for Download
                                            rate
    # y="Total System Power [W]",  # Replace with your actual column for System
                                             Power
    x="Total DL [MB]",
    y="Total System Power [W]",
    marker="o",
    markersize=8,
    color="blue",
    linewidth=2,
)
```

```
plt.title("NIC - Download Rate vs System Power")
plt.xlabel("Download Rate [KB/s]")
plt.ylabel("System Power (W)")
```

```
# Plot for Upload Rate vs System Power
plt.subplot(2, 1, 2)
sns.lineplot(
    data=df,
    # x="Current UP rate [KB/s]",  # Replace with your actual column for Upload
                                          rate
    # y="Total System Power [W]",  # Replace with your actual column for System
                                          Power
    x="Total UP [MB]",
    y="Total System Power [W]",
    marker="o",
    markersize=8,
    color="green",
    linewidth=2,
)
plt.title("NIC - Upload Rate vs System Power")
plt.xlabel("Upload Rate [KB/s]")
plt.ylabel("System Power (W)")
# Adjust layout
plt.tight_layout()
# Show plots
plt.show()
```

C.6 plot for memory usage vs system power consumption

```
 df = pd.read_csv(data_file)

# Group by battery percentage and calculate average statistics for memory load and
                                        power
grouped_data = (
    df.groupby("Battery_Percentage")
    .agg(
        avg_memory_load=("Physical Memory Load [%]", "mean"),
        avg_power=("Total System Power [W]", "mean"),
    )
    .reset_index()
)
```

```
# Set up the matplotlib figure
plt.figure(figsize=(7, 5))

# Plot average memory load vs average system power
sns.lineplot(
    data=grouped_data,
    x="avg_memory_load",
    y="avg_power",
    marker="o",
    label="System Power Cosumption",
    color="blue",
)
plt.title("Physical Memory Usage vs. System Power Consumption")
plt.xlabel("Physical Memory Usage (%)")
plt.ylabel("Average System Power (W)")
plt.grid(True)
plt.legend()

# Adjust layout
plt.tight_layout()

# Show plot
plt.show()
```

C.7 plot for cpu usage vs system power consumption

```
df = pd.read_csv(data_file)

# Group by battery percentage and calculate average statistics for CPU usage and
                                            power
grouped_data = (
    df.groupby("Battery_Percentage")
    .agg(
        avg_cpu_usage=("Total CPU Usage [%]", "mean"),
        avg_power=("Total System Power [W]", "mean"),
    )
    .reset_index()
)

# Set up the matplotlib figure
plt.figure(figsize=(7, 5))

# Plot average CPU usage vs average system power
sns.lineplot(
    data=grouped_data,
    x="avg_cpu_usage",
    y="avg_power",
    marker="o",
    label="Average System Power",
    color="blue",
)
plt.title("CPU Usage vs. System Power Consumption")
plt.xlabel("Average CPU Usage (%)")
plt.ylabel("Average System Power (W)")
plt.grid(True)
plt.legend()
# Adjust layout
plt.tight_layout()
# Show plot
plt.show()
```