ISSN 2355-3286

Enhancing Energy Visibility: A Real-Time Power Monitoring System via Web Scraping and NoSQL

Moeljono Widjaja[1], Rahmi Andarini[2]
[1] Department of Informatics, Universitas Multimedia Nusantara, Tangerang, Indonesia
[2] Department of Engineering Physics, Tangerang, Indonesia
moeljono.widjaja@umn.ac.id
rahmi.andarini@lecturer.umn.ac.id

Accepted March 11, 2026
Approved April 09, 2026

***Abstract*— Modern industrial power meters are often equipped with Ethernet ports, providing web-based interfaces for real-time monitoring. However, automating data extraction from these interfaces for long-term analysis can be challenging without costly proprietary software. This paper presents a systematic design for an energy logging system that utilizes a web scraping approach to systematically retrieve data from power meter web APIs. The system is implemented using a Python-based scraping script deployed on a Raspberry Pi and utilizes a MongoDB database for scalable data storage and subsequent load profile visualization. Experimental results from an installation at the Universitas Multimedia Nusantara campus demonstrate the system's ability to effectively monitor high-load Mechanical Ventilation and Air Conditioning (MVAC) systems, providing critical insights into peak consumption periods and operational efficiency. This low-cost, automated solution facilitates data-driven energy management and supports institutional energy-saving initiatives.**

***Index Terms*— Energy Monitoring, Load Profiles, TCP/IP Protocol, MongoDB, Web Scraping**

I. Introduction

The global energy crisis has become a crucial issue with significant economic and political consequences. Driven by rapid population growth and industrial expansion, energy demand continues to outpace fossil fuel reserves. In Indonesia, this issue is intensified by a heavy reliance on subsidized fuels, which often encourages inefficient consumption. Consequently, prioritizing energy efficiency—particularly in the commercial and construction sectors, where savings potential is estimated at 20% to 35% [1]—is critical.

Universitas Multimedia Nusantara (UMN) has actively committed to campus-wide efficiency, focusing specifically on its Mechanical, Ventilation, and Air Conditioning (MVAC) system, the university's significant energy user [2], [3]. Currently, Buildings C and D share a unified MVAC infrastructure, which prevents facility managers from isolating individual building performance. While separate meters exist for lighting, cooling consumption is recorded only via two shared chiller meters. This lack of granular data makes it difficult to identify waste or quantify the impact of specific conservation efforts [4].

The technical complexity of UMN's water-based chiller system further complicates monitoring. Water enters the chillers at 12°C and is cooled to 5–7°C before distribution to Fan Coil Units (FCUs) that serve clusters of three to four classrooms. While air at the FCU coil begins at 15°C and is initially distributed at 19°C, the integration of Fresh Air (FA) piping results in a final ambient classroom temperature of 25°C. This cooling process often runs at full capacity regardless of need; UMN's room occupancy averages only 53% to 64.3%, meaning chillers frequently cool unoccupied spaces. As noted by Zhang [5], the primary barrier to effective campus energy management is the combination of outdated monitoring technology and insufficient data granularity, which diminishes the sense of accountability among academic units [6]–[8].

The Internet of Things (IoT) offers a viable solution to these institutional challenges by providing a network of devices capable of remote monitoring and control. It has seen rapid adoption across multiple domains—notably in smart cities [9], [10], homes [11], and security [12]. IoT technology is increasingly attractive for smart campus management due to its ease of implementation [13], [14], though the field still lacks standardized application references for widespread methodological use [15].

To bridge this visibility gap, this paper proposes an automated energy logging system designed for real-time MVAC monitoring at UMN. The system utilizes

Python-based web scraping [16], [17] to extract data from industrial power meters, effectively bypassing the limitations of proprietary interfaces. By centralizing this data in a scalable MongoDB database, the framework establishes a technical foundation for data-driven management. The entire pipeline—from acquisition and processing to visualization—is integrated within a cohesive Python environment to ensure long-term scalability and efficiency.

II. METHODOLOGY

The research method involves several stages, namely: setting up the network configuration on the power meter and scraping data from the website

A. Setup Network Configuration and Web-Page Access

As explained in [18], the PowerLogic PM5560 series is equipped with an Ethernet port with the following default configuration:

- IP method = Stored
- IP address = 169.254.0.10
- Subnet mask = 255.255.0.0
- Gateway = 0.0.0.0
- HTTP server = Enabled
- DPWS = Enabled
- EtherNet/IP = Enabled
- DNP3 = Disabled
- Device name = PM55-#xxxxxxxxxx

The display of default network configuration is shown in **Error! Reference source not found.**.

For meters equipped with a display, basic Ethernet settings can be configured using the display. For meters without a display, it requires the following steps to configure basic Ethernet settings before connecting the meter to the local network.

1) Disconnect the computer from the network. If the computer has wireless communication, be sure to disable the wireless network connection.

NOTE: After you disconnect your computer from the network, its IP address should automatically update to a default IP address of 169.254.###.### (where ### equals a number from 0 to 255) and a subnet mask of 255.255.0.0.

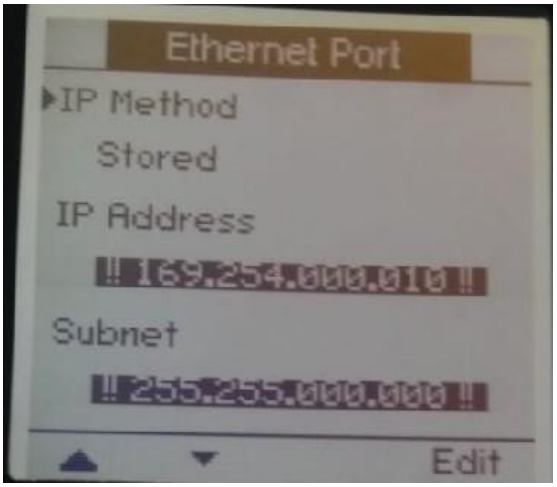

Fig. 1. Default Ethernet port configuration on the PowerLogic PM5560 series

TABLE I. DEFAULT LOGIN CREDENTIALS

Username	Password
user1	pass1
user2	pass2

2) Use an Ethernet cable to connect the computer to one of the meter's Ethernet ports.

3) Open a web browser and enter 169.254.0.10 in the address bar.

4) Log in to the meter webpage. The default login credentials can be found in Default Ethernet port configuration on the PowerLogic PM5560 series

5) .

The MAC address of the PowerLogic PM5560 installed on Chiller 2 is MAC: 00-80-67-AA-C4-45, as shown in **Error! Reference source not found.**.

An attempt was made to access the PowerLogic PM5560 installed in Chiller 2 in the Utility Building on the UMN campus via the web. Logging in to the web server was done using the default username and password. The logging process was successful; however, the webpage only displayed static data. Fig. 3 shows the web-based configuration interface for a Schneider Electric PowerLogic PM5560, which is a high-end power and energy meter used in industrial and commercial electrical systems. The page appears to be missing its CSS (styling). Instead of a professional dashboard with buttons and a sidebar, it is displaying as a plain list of blue hyperlinks on a white background. This often happens if:

- The connection is slow or unstable.
- The browser blocked the loading of scripts/styles for security reasons (common with "Not Secure" HTTP connections).
- The meter's internal web server is under high load

Fig. 2. MAC address on PowerLogic PM5560 installed on Chiller 2

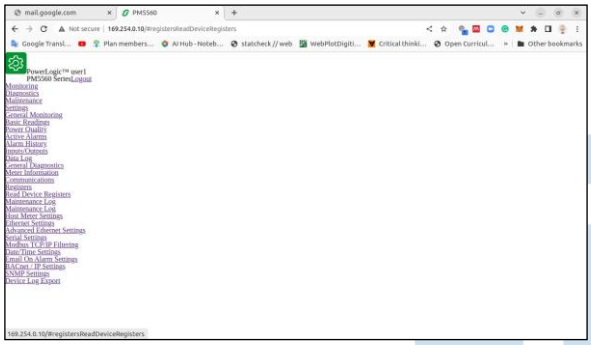

Fig. 3. Webpage on PowerLogic PM5560 Chiller 2

However, after restarting the Power Meter unit, the web server can display a complete dashboard as shown in **Error! Reference source not found.**. Now, it shows the fully rendered web interface for a Schneider Electric PowerLogic PM5560 Series power meter. Unlike the previous image, the styling (CSS) is loading correctly, providing a graphical dashboard for real-time electrical monitoring.

Here is an overview of the dashboard as shown in **Error! Reference source not found.**.

- Device Identity: The meter is a PM5560 Series, accessed via IP 192.168.11.254.
- User: Logged in as teknisi1.
- Active View: The user is in the Monitoring tab under General Monitoring - Basic Readings.

The dashboard is split into two primary visual areas:

1) Gauges (Visual Representation): This section provides a quick look at the Load Current for the three phases:

a) Phase A (Ia): 280.39 A.
b) Phase B (Ib): 322.11 A.
c) Phase C (Ic): 267.51 A.
d) Users can toggle the radio buttons to switch these gauges to display Power, Voltage LL (Line-to-Line), or Voltage LN (Line-to-Neutral).

2) Basic Readings Table (Detailed Data): This table provides a comprehensive snapshot of electrical performance, including Minimum, Present, and Maximum recorded values:

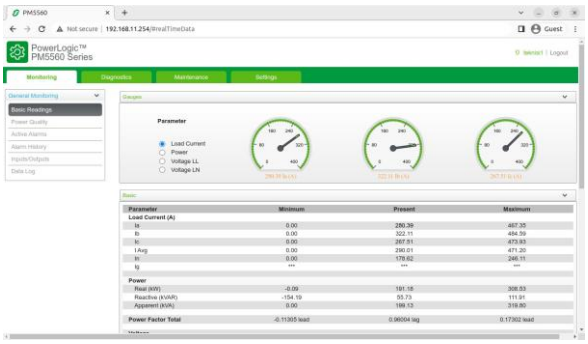

Fig. 4. Webpage on PowerLogic PM5560 Chiller 2 after restarting the Power Meter unit

a) Load Current (A): Shows the current for each phase, the average (I_{Avg}), and the neutral current (I_n).
b) Power (kW, kVAR, kVA):
- Real Power: Currently 191.18 kW.
- Reactive Power: Currently 55.73 kVAR.
- Apparent Power: Currently 199.13 kVA

c) Power Factor Total: Currently 0.96004 lag. A "lagging" power factor usually indicates an inductive load, such as motors or transformers

B. Web-Scraping Data PowerLogic PM5560

After successfully accessing the website from the PowerLogic PM5560, the next step is to automatically read the required parameters by performing web-scraping.

To perform web scraping from the PowerLogic PM5560 website, do the following:

1) Open the PM5560 website on your local network.
2) Log in to the website using username = user1 and password = pass1.
3) Inspect the webpage by right-clicking and selecting "Inspect".
4) Click the "Network" tab, then refresh by pressing the "Clear" button.
5) Select an element on the webpage to inspect by pressing "ctrl-shift-C."
6) Copy the section of the element to be extracted as "Copy as cURL" (see **Error! Reference source not found.**).
7) Paste the "cURL command" into the website https: //curlconverter.com/python/
8) Copy the converted file to Python by clicking "Copy to clipboard."

By combining these steps, a technician or engineer can "sniff" the communication between the browser

and the PM5560 meter. The resulting cURL command can then be pasted into a terminal or a script to programmatically retrieve the meter's current, power, and voltage readings for external logging or custom dashboards.

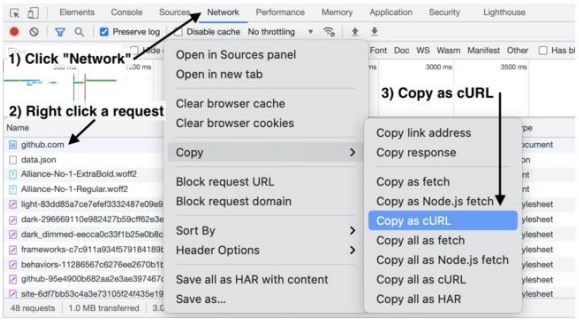

Fig. 5. Illustration of the web-scraping process

Based on the web inspection, identify key features to fetch useful data from the website. An example of a Python script obtained from the web-scraping can be seen in Listing 1.

This Python script is a practical application of the "Copy as cURL" technique shown in **Error! Reference source not found.**. It automates the process of requesting real-time data directly from the Schneider Electric PM5560 power meter.

Listing 1. Python script for web-scraping

```
  import requests
  from bs4 import BeautifulSoup

  headers = {
      'Accept': 'text/plain, /; q=0.01',
      'Accept-Language':'en-
US,en;q=0.9,id;q=0.8',
      'Authorization':'Basic
***************',
      'Connection': 'keep-alive',
      'Content-Type':'application/x-www-form-
urlencoded; charset=UTF-8',
      'Origin': 'http://192.168.11.254',
      'Referer': 'http://192.168.11.254/',
      'User-Agent':    'Mozilla/5.0    (Linux;
Android 6.0; Nexus 5 Build/MRA58N)
      AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/107.0.0.0 Mobile
      Safari/537.36',
      'X-Requested-With': 'XMLHttpRequest',
  }

  data=
'PL_*^H3000[86]PL,PL*^H3110[2]PL,PL*^H3200[48
]PL,PL*^H3706[94]PL,
  PL*^H3802[54]PL,PL*^H27218[96]PL,PL*^H27616
[2]PL,PL*^H27694[96]PL,
  PL*^H28092[2]_PL'

  response=
requests.post('http://192.168.11.254/UE/Post_
PL_Data',
  headers=headers, data=data, verify=False)
```

The detailed explanation of Listing 1 is given in the following:

1) Script Components

- Target Device: The script sends a request to 192.168.11.254, which matches the IP address of the PM5560 meter seen in the web interface.
- Authentication: The 'Authorization': 'Basic **********' header handles the login credentials for the user (e.g., teknisi1) so the script can access the data without a manual login.
- Request Method: It uses requests.post to hit the /UE/Post_PL_Data endpoint, which is the internal "mailbox" the meter uses to serve data to the web dashboard.

2) The Data Payload (Modbus Registers) The data string contains the core instructions for what information to retrieve:

- Register Groups: Codes like H3000, H3110, and H3200 correspond to specific Modbus registers within the meter.
- Requested Parameters: These registers contain the values you see on the dashboard, such as Load Current, Power (kW), and Voltage.
- Syntax: The PL_*^...PL format is a proprietary Schneider Electric wrapper used to batch multiple register requests into a single network call for efficiency.

3) Purpose and Outcome

- Bypassing the GUI: Instead of looking at the visual gauges and tables, this script pulls the raw numbers directly into Python.
- Data Integration: Once the response is received, a developer can use BeautifulSoup to parse the text and save it to a database, an Excel file, or a custom monitoring system.
- Scalability: This allows for continuous logging of parameters like the 0.96004 lag Power Factor or 322.11 A Load Current at high frequencies (e.g., every second).

In order to streamline the process of logging, the parsing process is modularized as a function as described in Listing 2. To parse the response from the Schneider Electric PM5560, you can process the raw text returned by the meter into a structured Python dictionary. Since the meter returns data in a proprietary string format (delimited by PL_ and _PL markers), we can use a helper function to clean and map these values to the electrical parameters seen on the dashboard.

This method is very effective due to the following reasons:

- Automation: This script allows logging the 322.11 A current or 191.18 kW power readings every few seconds without manually refreshing a browser.

- Data Integrity: By targeting the specific registers (like H3000), for pulling the exact numerical value stored in the meter's memory.
- Scalability: easy expansion of the payload string to include more registers, such as Voltage LL or Harmonics, by adding their respective Modbus addresses.

Listing 2. Parsing raw data

```
def requestData():
    data = dict()
    data_raw=
'PL__*^H3000[86]__PL,PL__*^H3110[2]__PL,PL__*
^H3200[48]__PL,PL__*^H3706[94]__PL,PL__*^H380
2[54]__PL,PL__*^H27218[96]__PL,PL__*^H27616[2
]__PL,PL__*^H27694[96]__PL,PL__*^H28092[2]__P
L'
    response=
requests.post('http://192.168.24.20/UE/Post__
PL__Data',  headers=headers,  data=data_raw,
verify=False)
    data_text = response.text
    data_lst = data_text.split(',')
    data_int = list(map(int,data_lst))
    data['Ia'] = toFL32(data_int[0:2])
    data['Ib'] = toFL32(data_int[2:4])
    data['Ic'] = toFL32(data_int[4:6])
    data['Vab'] = toFL32(data_int[20:22])
    data['Vbc'] = toFL32(data_int[22:24])
    data['Vca'] = toFL32(data_int[24:26])
    data['P'] = toFL32(data_int[60:62])
    data['pf'] = toFL32(data_int[84:86])
    data['freq'] = toFL32(data_int[86:88])
    A = data_int[92]
    B = data_int[93]
    C = data_int[94]
    D = data_int[95]
    data['E']=
(A*281474976710656+B*4294967296+C*65536+D)/10
00
    return data
```

In the payload string 'PL_*^H3000 [86]PL. . . ' , the number in brackets (e.g., [86]) tells the meter to read 86 consecutive registers starting from address 3000.

To extract a specific value, the position must be specified in the response string such as:

- H3000 will contain I_a.
- H3002 will contain I_b.
- H3004 will contain I_c.

This Python script is a complete Internet of Things (IoT) Data Logger. It bridges the gap between the physical hardware (the PM5560 Power Meter) and a cloud database (MongoDB).

Listing 3. Setting connection to MongoDB database

```
WIB = tz.gettz('WIB')
MONGODB_SRV                                    =
'mongodb+srv://username:***********@energy.gs
1cq.mongodb.net/test?retryWrites=true&w=major
ity'
MONGODB_DB = 'SmartEnergy'
MONGODB_COLLECTION = 'PowerMeter'

loggerID = 'CHILLER_2'
```

The data obtained through web scraping is register data of type Integer. The current, voltage, power, power factor, and frequency data are stored in two registers that must be converted to Float32 format. The conversion process from two Integer numbers to Float32 is performed using the function shown in Listing 4.

Listing 4. Converting to Float32

```
def toFL32(val):
    a = val[0]
    b = val[1]
    c=struct.unpack('>f',
bytes.fromhex(f"{a:0>4x}" + f"{b:0>4x}"))[0]
    return c
```

Power meters store data in 16-bit registers, but values like "Amps" or "kW" are usually 32-bit floating-point numbers. Therefore, they must be converted using a conversion function (toFL32) as given in Listing 4. Here is the explanation of the conversion function:

- The Problem: The meter sends two separate 16 -bit integers for one reading.
- The Solution: The toFL32 function takes those two integers (val[0] and val[1]), converts them to Hex, joins them together, and uses struct. unpack to turn them back into a human-readable decimal (Float 32).

3) Database Integration (logData) This function acts as the "courier":
   - Internet Check: It pings Google/MongoDB to ensure the connection is live.
   - MongoDB Upload: It connects to a cluster named SmartEnergy and inserts a new document into the PowerMeter collection.
   - Metadata: Each entry is tagged with loggerID: 'CHILLER_2', allowing you to distinguish this meter's data from others in the same database.

The complete source code is shown in Listing 5.

Listing 5. Logging Data to MongoDB Database

```
def logData():
    url = "https://cloud.mongodb.com/"
    timeout = 5
    my_dict = {'loggerID' : loggerID}
    my_dict.update(readTimeStamp())
    my_dict.update(requestData())
    print(my_dict)
    try:
        request      =      requests.get(url,
timeout=timeout)
        print("Connected to the Internet")

        try:
            client = False
            client                             =
pymongo.MongoClient(MONGODB_SRV)
            print("MongoDB:   connection   is
opened")
            db=client[MONGODB_DB]
            collection                         =
db[MONGODB_COLLECTION]
            try:
```

```
                result                          =
collection.insert_one(my_dict)
                print ("MongoDB: successfully
insert data.")
            except
pymongo.errors.NetworkTimeout as error:
                print("MongoDB    error:    ",
error)
            except
pymongo.errors.ServerSelectionTimeoutError as
error:
                print("MongoDB    error:    ",
error)

        except
pymongo.errors.ConfigurationError as error:
            print("MongoDB error: ", error)
        finally:
            if client:
                client.close()
                print("MongoDB: connection is
closed")
    except          (requests.ConnectionError,
requests.Timeout) as exception:
        print("No internet connection.")
```

The sent energy data consists of four registers and must be combined into one value with a calculation that can be seen in Listing 6. The energy *E* is stored across four 16-bit registers: *A, B, C, D*. The script reconstructs the 64-bit value and scales it to kilowatt-hours (kWh).

The mathematical expression used in the code is:

$$E = \frac{(A \cdot 2^{48}) + (B \cdot 2^{32}) + (C \cdot 2^{16}) + D}{1000}$$

Note: the code uses 281,474,976,710,656 which is 2^{48} and 4,294,967,296 which is 2^{32}.

Listing 6. Converting four registers into one value

```
A = data_int[92]
B = data_int[93]
C = data_int[94]
D = data_int[95]
data['E']                                   =
(A*281474976710656+B*4294967296+C*65536+D)/10
00
```

Instead of running once and stopping, the script uses the *timeloop* library in order to regularly logging data from the power meter. Listing 7 implements the periodic task. The @tl.job decorator is set to run the logData() function every 5 minutes.

Listing 7. Executing periodic task

```
from timeloop import Timeloop

tl = Timeloop()

@tl.job(interval=timedelta(minutes=5))
def sample_job_every_5m():
    logData()
    print  ("Finish  5m  job  current  time  :
{}".format(time.ctime()))
```

After successfully web-scraping data from the PowerLogic PM5560 and converting the register data into data that matches the measured parameters, the next step is to periodically send the data to the MongoDB Atlas database via the Internet.

III. Results and Discussion

This section will explain two folds: the results of study the electrical system and logging the power meter via web scraping, and the discussion on the visualization of the logged data.

A. The Electrical System Study on the UMN Campus

This section will explain the electrical data for Buildings C and D on the Multimedia Nusantara University campus. The UMN campus has four main buildings, as shown in **Error! Reference source not found.**.

Fig. 6. Main building on yhe UMN Campus
Source: https://360.umn.ac.id/

The distribution of the kWh meter panels can be seen in **Error! Reference source not found.**. The electricity panel Chiller 1 kWh meter is used to measure all electricity consumption in the Utility Building and Chiller 1. Electricity consumption in the Utility Building consists of Chiller plant I, including pumps, fans, and cooling towers, as well as all lights in the Utility Building. The kWh Chiller 1 also reads the electricity usage of the Building C transfer pump, which is the pump that functions to move water from the ground tank to the roof tank. The Building C transfer pump usually runs 2 to 3 times a day. Chiller 1 is only used 1 to 2 times a week, namely on weekends or in the afternoon when the cooling load is not too large.

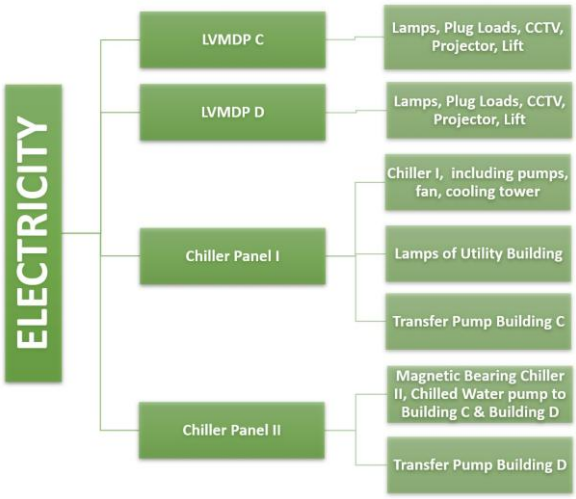

Fig. 7. Division of kWh meter panels in Buildings C and D

The electricity Panel Chiller 2 kWh meter reads the electrical energy consumption of Chiller Plant 2, including the pumps that distribute cold water to Building C and Building D. Chiller 2 in question is a magnetic bearing type with a high level of efficiency used routinely in Buildings C and D during working days and working hours. In addition, the electricity consumption of the transfer pump in Building D is also recorded on the electricity kWh meter in Panel Chiller 2.

Given the combined use of the Chiller between Buildings C and D, the development of this monitoring system is expected to determine the amount of energy consumed by Building C, particularly for the use of the Chiller.

B. Data Visualisaion

Installation and testing of the Modbus logger have been successfully carried out for the PowerLogic PM800 Power Meter, while readings from the PowerLogic PM5560 were done by web-scraping on the website available on the unit.

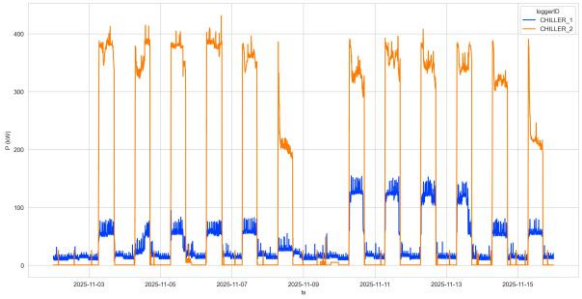

Fig. 8. Profiles of power consumptions from Chiller 1 and 2 including LVMDP building C and D

Fig. 8 shows a plot of the Power Meter data reading results for Chiller units 1 and 2 and LVMDP buildings C and D. During peak load (morning to afternoon) the power consumption of Chiller 2 is highest and the lowest is Chiller 1, while the power consumption in Buildings C and D is relatively the same. Based on information from the building management, Chiller 2 is newer and more efficient so that the operation of Chiller 2 is prioritized over Chiller 1.

1) *Daily Electrical Energy Load Profile Analysis:* The electrical energy load profile of the chiller for a single weekday can be seen in Fig. 9. Peak load occurs between 7 a.m. and 5 p.m., with the largest energy load on Chiller 2. This incident is in accordance with the MVAC system schedule at the university that Chiller 2 is the main Chiller operated during weekdays. There is a sudden increase in load at the start of the day, reaching 350 kW, as the chillers are turned on at approximately 6 a.m. This schedule is implemented because it requires some time to cool the classrooms before the first-class schedule starts at 8 a.m. Most classes end at 4 p.m. There is an opportunity to shape the load between 7 a.m. and 4 p.m. by adjusting the chiller set point to match the load of classes conducted during the day. In addition, since it is known that the average Chiller's power approximately at 310 kW, then reducing its operating hours by 2 hours per day will reduce energy consumption by 620 kWh, or about 18 percent.

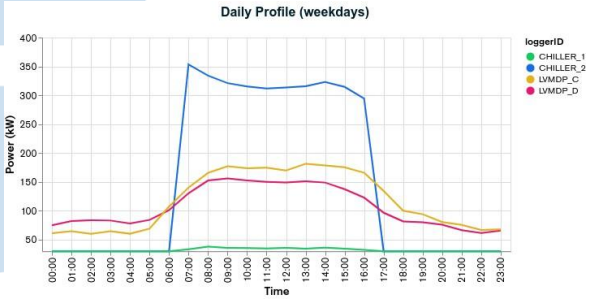

Fig. 9. Daily profiles (Monday-Friday) of load consumption from Chiller 1 and 2 including LVMDP building C and D

Fig. 10. Weekly profile of power consumption Chiller 1 and 2 and LVMDP of buildings C and D

2) *Weekly Electrical Energy Load Profile Analysis:* The electrical energy load profile for one week can be seen in Fig. 10. Peak load occurs during weekdays (Monday - Friday). The largest energy load is on Chiller 2. Most activities on the campus are conducted during weekdays, while there are a few classes also conducted on Saturday. The load reflects the activities on campus from Monday to Sunday. There is also an opportunity to optimize the load by

adjusting the chiller proportional to the number of classes held on specific days.

IV. Conclusion

This paper has presented the systematic design and successful implementation of an automated energy logging system that utilizes a web scraping approach to monitor industrial power meters. By leveraging the built-in web API of the Schneider Electric PowerLogic PM5560, the system provides a systematic method to bypass the limitations of manual data recording and expensive proprietary software, enabling continuous, high-frequency data acquisition. The integration of Python-based scraping with a MongoDB Atlas cloud database offers a scalable, low-cost, and robust IoT solution for institutional energy management.

The deployment at the Universitas Multimedia Nusantara campus provided critical granular visibility into the load profiles of the university's MVAC systems. Data analysis confirmed that peak electrical loads consistently occur on weekdays between 7:00 a.m. and 5:00 p.m.. Furthermore, the findings validated the campus operational strategy of prioritizing the more efficient magnetic-bearing Chiller 2 over Chiller 1. Crucially, the system identified significant opportunities for optimization, such as a potential 18% reduction in energy consumption through the adjustment of chiller operating hours. This level of transparency is a vital prerequisite for achieving the 20–35% energy savings potential identified in the commercial and construction sectors.

Future work will expand the system's scope by integrating additional meters across the campus and developing a real-time web dashboard for live visualization of energy performance indicators. Additionally, the accumulated historical data will serve as the foundation for developing predictive models for cooling load demand and automated anomaly detection, further enhancing the efficiency of building operations.

ACKNOWLEDGMENT

This research was supported by Universitas Multimedia Nusantara under contract no 058/PI/LPPM-UMN/III/2022.

References

[1] Menteri ESDM RI, "Handbook of Energy & Economy Statistics of Indonesia 2020," Book, pp. 1–111,2021. [Online]. Available:https://www.esdm.go.id/en/publication/ handbook-of-energy-economic-statistics-of-indonesia-heesi

[2] M. Zhou, A. M. Abdulghani, M. A. Imran, and Q. H. Abbasi, "Internet of things (iot) enabled smart indoor air quality monitoring system," ACM International Conference Proceeding Series, pp. 89–93, 2020. doi: 10.1145/3398329.3398342.

[3] R. Andarini, M. Widjaja, and A. M. Sutopo, "Penilaian kualitas udara dalam ruangan di kampus universitas multimedia nusantara," Technical Report, 2021.

[4] R. Andarini, M. Salehuddin, and C. O. Harahap, "Pengembangan Sistem Manajemen Energi di Kampus Universitas Multimedia Nusantara," Universitas Multimedia Nusantara, Tangerang, Tech. Rep., 12 2019.

[5] J.-P. Zhang, J. Zhou, Q.-M. Luo, and G.-B. Fan, "Research on Campus Energy Consumption Monitoring System Based on IPV6," DEStech Transactions on Social Science, Education and Human Science, no. icaem, 3 2018. doi: 10.12783/DTSSEHS/ICAEM2017/19145.

[6] K. A. Barfi, "Internet of things applications for smart environments," Intelligent Systems Reference Library, vol. 121, pp. 93–103, 2022. doi: 10.1007/978-3-030-97516-6 5.

[7] J. Saini, M. Dutta, and G. Marques, "Sensors for indoor air quality monitoring and assessment through Internet of Things: a systematic review," Environmental Monitoring and Assessment, vol. 193, no. 2, 2 2021. doi: 10.1007/S10661-020-08781-6.

[8] M. Widjaja, D. K. Halim, and R. Andarini, "The development of an iot-based indoor air monitoring system towards smart energy efficient classroom," Ultima Computing : Jurnal Sistem Komputer, vol. 14, pp. 28–35, 2022. doi: 10.31937/sk.v14i1.2565.

[9] J. Shah and B. Mishra, "IoT enabled environmental monitoring system for smart cities," 2016 International Conference on Internet of Things and Applications, IOTA 2016, pp. 383–388, 2016. doi: 10.1109/IOTA.2016.7562757.

[10] B. Hammi, R. Khatoun, S. Zeadally, A. Fayad, and L. Khoukhi, "IoT technologies for smart cities," IET Networks, vol. 7, no. 1, pp. 1–13, 1 2018. doi: 10.1049/IET-NET.2017.0163.

[11] L. Smirek, G. Zimmermann, and M. Beigl, "Just a Smart Home or Your Smart Home - A Framework for Personalized User Interfaces Based on Eclipse Smart Home and Universal Remote Console," Procedia Computer Science, vol. 58, pp. 107–116, 2016. doi: 10.1016/J.PROCS.2016.09.018.

[12] S. S. Vedaei, A. Fotovvat, M. R. Mohebbian, G. M. Rahman, K. A. Wahid, P. Babyn, H. R. Marateb, M. Mansourian, and R. Sami, "COVIDSAFE: An IoT-based system for automated health monitoring and surveillance in post-pandemic life," IEEE Access, vol. 8, pp. 188538– 188551, 2020. doi: 10.1109/ACCESS.2020.3030194.

[13] A. Zhamanov, Z. Sakhiyeva, R. Suliyev, and Z. Kaldykulova, "IoT smart campus review and implementation of IoT applications into education process of university," 2017 13th International Conference on Electronics, Computer and Computation (ICECCO), Abuja, Nigeria, 2017, pp. 1–4. doi: 10.1109/ICECCO.2017.8333334.

[14] G. Lilis, G. Conus, N. Asadi, and M. Kayal, "Towards the next generation of intelligent building: An assessment study of current automation and future IoT based systems with a proposal for transitional design," Sustainable Cities and Society, vol. 28, pp. 473–481, 1 2017. doi:10.1016/J.SCS.2016.08.019.

[15] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey," Computer Networks, vol. 54, no. 15, pp. 2787–2805, 10 2010. doi: 10.1016/J.COMNET.2010.05.010.

[16] C. Bhatt, A. Bisht, R. Chauhan, A. Vishvakarma, M. Kumar, and S. Sharma, "Web Scraping Techniques and Its Applications: A Review," Proceedings - 2023 3rd International Conference on Innovative Sustainable Computational Technologies, CISCT 2023, 2023. doi: 10.1109/CISCT57197.2023.10351298.

[17] M. A. Wahed, M. S. Alzboon, M. Alqaraleh, J. Ayman, M. AlBatah, and A. F. Bader, "Automating Web Data Collection: Challenges, Solutions, and Python-Based Strategies for Effective Web Scraping," 2024 7th International Conference on Internet Applications, Protocols, and Services, NETAPPS 2024, 2024. doi: 10.1109/NETAPPS63333.2024.10823528.

[18] E. Schneider, "PowerLogic PM5500 series user manual," 7 2020.[Online].Available:https://www.se.com/ww/en/download/document/ HRB1684301/