

ĐỒ ÁN 1

BÁO CÁO TIẾN ĐỘ TUẦN 12(15/3/2025 – 21/3/2025)

Nguyễn Thanh Duy- 2210527

Yêu cầu: thêm một số tính năng trên source code có sẵn

Công việc đã hoàn thành
- Hiển thị, căn chỉnh bảng 6x6 hiển thị giá trị trung bình được gửi về từ cảm biến ở Tracking
- Chỉnh sửa phần Logs, cho phép người dùng có thể xem lịch sử bằng việc ẩn mũi tên

I/ Ý tưởng để

1/ Hiển thị giá trị trung bình của cảm biến đọc từ file .log hiển thị dưới dạng bảng tương ứng với vị trí cảm biến, cảm biến nào đang được ghi sẽ tô sáng.

- Dùng regex để đối chiếu giá trị đọc từ file log
- Tính giá trị trung bình đổi sang điện áp
- Vẽ bảng hiển thị, mỗi khi có một cảm biến được đọc vào thì nó sẽ gán màu cho cặp đó và vẽ lại màn hình để hiển thị màu.
- Tạo 1 luồng để nó liên tục cập nhật mà không ảnh hưởng khi ta đang làm việc với cửa sổ khác

2/ Cuộn cửa sổ bằng phím mũi tên:

- Dựa vào kích thước terminal để giới hạn số lượng lịch sử gần đây nhất của các lệnh được hiển thị
- Dùng 2 file để lưu lệnh, 1 file lưu chính và 1 file lưu tạm, khi gõ lệnh nó sẽ lưu vào cả 2 file, file tạm đó để dùng hiển thị ra cửa sổ vì khi có lệnh clear hay là khi ta khởi động lại chương trình thì file tạm sẽ trống, hiển thị cửa sổ sạch.
- Dùng 1 biến current_line dựa trên file tạm để làm con trỏ xác định vị trí.
- Nếu ấn mũi tên lên thì nó sẽ nhận event phím và thay đổi cái vị trí bắt đầu và kết thúc dựa vào kích thước terminal và dựa vào cái biến current_line được lưu khi nhập lệnh để biết bắt đầu và kết thúc hiển thị, từ đó vẽ lại cửa sổ.
- Tương tự ấn mũi tên xuống thì nó cũng dựa vào cách đó thay đổi vị trí bắt đầu và kết thúc để vẽ lại.
- Lưu ý giới hạn nếu ta đạt giới hạn trên hay giới hạn dưới thì khi ấn mũi tên sẽ không có tác dụng.

II/ Code:

1/ Bảng Trackings

```
#Tính giá trị trung bình, xét điều kiện lớn hơn 2 là do bỏ 2 giá trị đầu, kiểm tra những số khác 0 và chỉ lấy trung bình những số đó
def DAC_avarage (self):
    a = 0
    sum = 0
    for i,value in enumerate(self.output_log_line):
        if (i > 1) and (value != 0):
            sum += value*3.3/16383
            a+=1
        continue
    return f"{sum/a:.2f}"

#Tiền xử lý khi đọc từ file.log về
def process_data(self,log_line):
    patern = r"C(\d+)-(\d+)\\"
    r".*?[T: 0\]-V[ADC: (\d+)\]"
    r"(?:.*?[T: 1\]-V[ADC: (\d+)\])?"
    r"(?:.*?[T: 2\]-V[ADC: (\d+)\])?"
    r"(?:.*?[T: 3\]-V[ADC: (\d+)\])?"
    match = re.search(patern, log_line)
#patern ở đây là 1 regex dùng để đối chiếu với 1 hàng trong file log, do mỗi lần ta đọc về từ file.log là ta đọc 1 hàng
#thì ở đây nó trả về 6 cái (\d+) bao gồm hàng, cột, lần lấy 0,1,2,3
#match = re.search() sẽ lưu nó vào matchmatch

    if match:
        self.output_log_line = list(match.groups()) #match.groups() là lấy tất cả những gì có trong match, list() là lưu nó vào 1 biến mới dưới dạng list
        self.output_log_line = [0 if x is None else x for x in self.output_log_line]
        self.output_log_line = list(map(int,self.output_log_line))
        self.sensor_value[self.output_log_line[0]-1][self.output_log_line[1]-1] =
self.DAC_avarage()
        #return self.sensor_value
        self.format_table(self.output_log_line[0],self.output_log_line[1])

    else:
        print("Không tìm thấy dữ liệu cảm biến!")

def send_to_matrix(self):
    with open("test.log", "r", encoding="utf-8") as file:
        for log_line in file:
            if "ADC" in log_line: #kiểm tra trong dòng log đó có kí tự nào là ADC k, nếu có thì cho phép đọc dòng đó vì đó là dòng có giá trị cảm biến
                self.process_data(log_line)
                time.sleep(1)
                #app.invalidate()

def format_table(self,x,y):
    self.formatted = []
```

```

#self.sensor_value = [[str(col).ljust(15) for col in row] for row in self.sensor_value]
table_str = tabulate(self.sensor_value,tablefmt="grid",floatfmt=".2f").split("\n")
semaphore = 0
row_indx, col_indx = 0,0
for i,row in enumerate(table_str):
    if i % 2 == 0:
        for char in row:
            self.formatted.append(("fg:white",char))
            self.formatted.append(("fg:white","\n"))
    if i % 2 != 0:
        col_indx = 0
        semaphore = 0
        for char in row:
            if char.isdigit() or char == "." or char == " ":
                if row_indx == (x-1) and col_indx == (y-1):
                    self.formatted.append(("bg:yellow",char))
                else:
                    self.formatted.append(("fg:green",char))
            if char == "|":
                self.formatted.append(("fg:white",char))
                semaphore +=1
                if semaphore > 1:
                    col_indx +=1
            # else:
            #     self.formatted.append(("fg:white",char))
            self.formatted.append(("fg:white","\n"))
        row_indx +=1
return self.formatted

```

2/ Phần mũi tên lên(xem lịch sử trước)

```

def setup_keybindings(self):
    #viết một hàm với sự kiện là nó sẽ thực hiện nếu ta bấm phím mũi tên lênlên
    @self.kb.add('up')
    def _(event):

        if self.mode == 'menu':
            selectable_count = sum(1 for item in self.menu_items if not self.is_divider(item))
            self.selected_item = max(0, self.selected_item - 1)
            self.lasted_selected_item = self.selected_item
            selectable_items = self.get_selectable_items()
            self.info_frame.title = selectable_items[self.selected_item]

        elif self.mode == 'info':
            selectable_items = self.get_selectable_items()

            #Hàm xử lý cuộn lịch sử xem trên cửa sổ logs
            if selectable_items[self.selected_item] == "Logs": #nếu đang ở Logs, mũi tên dùng
để cuộn

```

```

        if self.current_line != 0:
            os_line = os.get_terminal_size().lines
            #Dùng size terminal để giới hạn vị trí
            #bắt đầu- kết thúc hiển thị ra cửa sổ
            self.current_line -= 1
            if self.save_current_line < os_line - 12:
                #Kiểm tra xem nếu dòng hiện tại bé
                #hơn kích thước terminal thì ta bù lại cho nó giữ trạng thái cũ
                self.current_line += 1
                #Lí do là để nếu lỡ số dòng đang nhỏ hơn
                #kích thước cửa mà ta ấn mũi tên lên thì khi tính start_idx nó ra số âm gây lỗi
            if self.save_current_line >= os_line - 12:
                if self.current_line < os_line - 12:
                    #Lí do os_line - 12 là do số hàng của
                    #phần cửa sổ hiển thị bằng kích thước hàng terminal - 12, 12 là các dòng hiển thị mấy cửa sổ
                    #này kia khác
                    self.current_line = os_line - 12
                    #Giới hạn lại vị trí dòng so với cửa sổ
                    #terminal, nếu có giảm hơn thì đặt nó về lại cái giới hạn đó thôi
                if self.current_line >= os_line - 12:
                    #Các lệnh dưới đây mục tiêu là ghi các
                    #lệnh được giới hạn bởi các điều kiện ở trên vào 1 list rồi khi vẽ lại giao diện thì nó hiện ra
                    #thoithoải

                    if self.current_line - (os_line - 12) == 0:
                        start_idx = 0
                    else:
                        start_idx = self.current_line - (os_line - 12)
                    self.create_info_log_cmd = ""

                    for i in range(start_idx, start_idx + (os_line - 12)):
                        self.create_info_log_cmd += self.create_info_log_raw[i]
                    self.create_info_log = None
                    self.create_info_log = self.create_info_log_cmd

```

3/ Phần mũi tên xuống(xem lịch sử gần đây)

```

@self.kb.add('down')
def _(event):

    if self.mode == 'menu':
        selectable_count = sum(1 for item in self.menu_items if not self.is_divider(item))
        self.selected_item = min(selectable_count - 1, self.selected_item + 1)
        self.lasted_selected_item = self.selected_item
        selectable_items = self.get_selectable_items()
        self.info_frame.title = selectable_items[self.selected_item]

    elif self.mode == 'info':
        selectable_items = self.get_selectable_items()

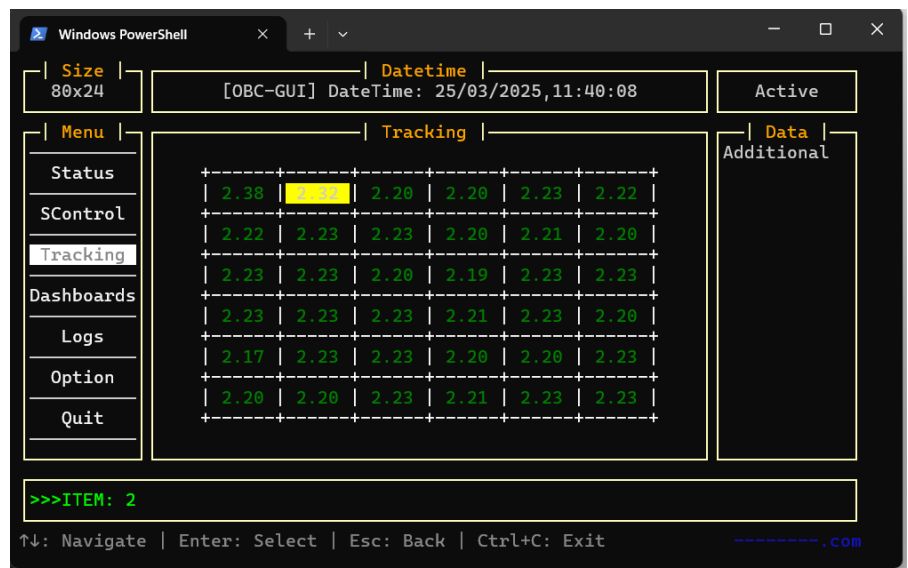
        #Hàm xử lý cuộn xem lịch sử cửa sổ logs
        if selectable_items[self.selected_item] == "Logs":
            if self.current_line != 0:
                self.current_line += 1
                #Khi ấn mũi tên xuống thì tăng dòng,
                #để giới hạn vị trí bắt đầu- kết thúc hiển thị trong cửa sổ đó
                if self.current_line > self.save_current_line:
                    #Đặt lại dòng nếu ta có ấn
                    #nhiều lần lớn hơn thì đặt nó về lại
                    self.current_line = self.save_current_line
                if self.current_line <= self.save_current_line:
                    #Từ việc biết dòng thì ta sẽ
                    #chặn vị trí bắt đầu và kết thúc để hiển thị khoảng đó ra cửa sổ

```

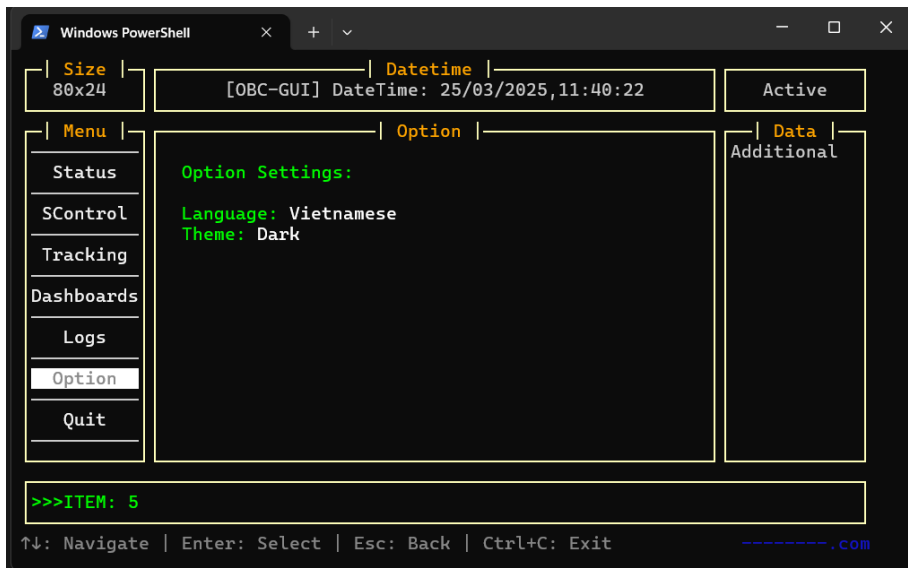
```
os_line = os.get_terminal_size().lines
start_idx = max(0, self.current_line - (os_line - 12))
self.create_info_log_cmd = ""
for i in range(start_idx, self.current_line):
    self.create_info_log_cmd += self.create_info_log_raw[i]
self.create_info_log = None
self.create_info_log = self.create_info_log_cmd
```

Kết quả

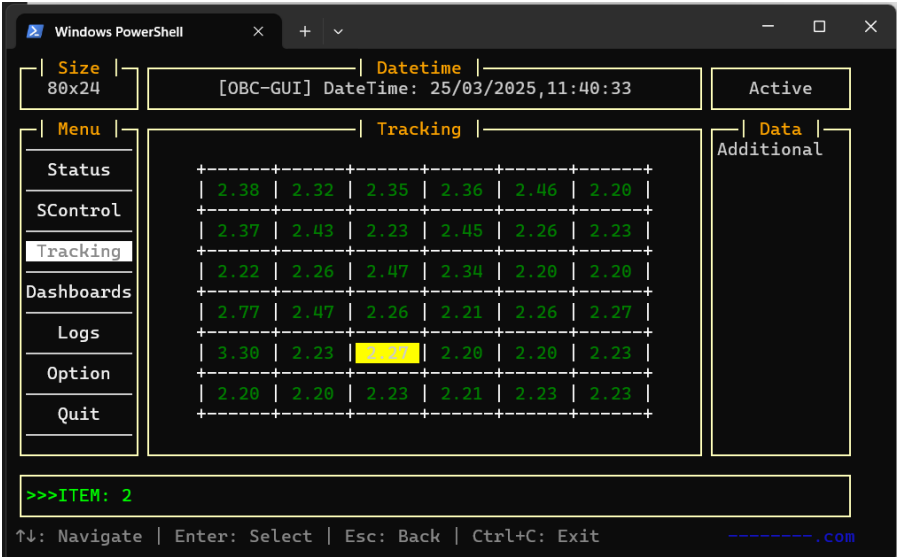
1/ Cửa sổ bảng hiển thị giá trị cảm biến



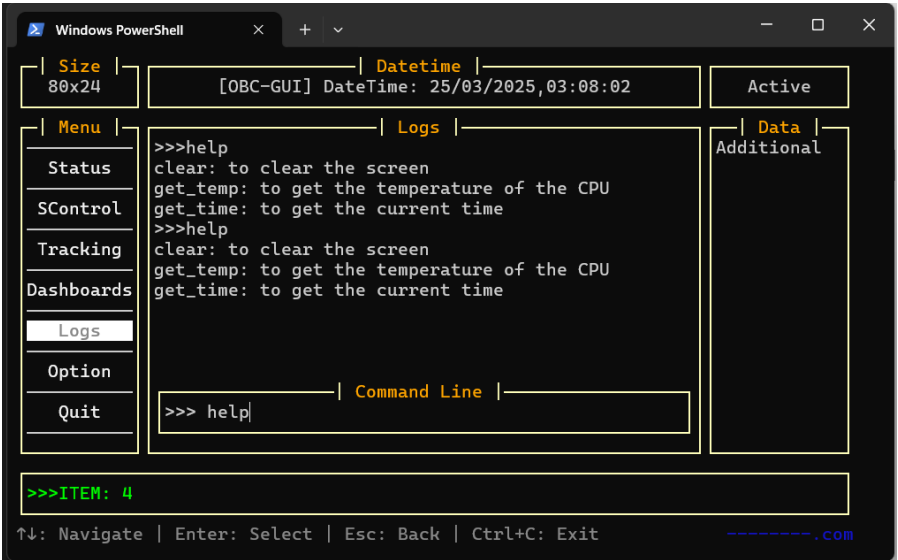
Thử chuyển sang cửa sổ khác



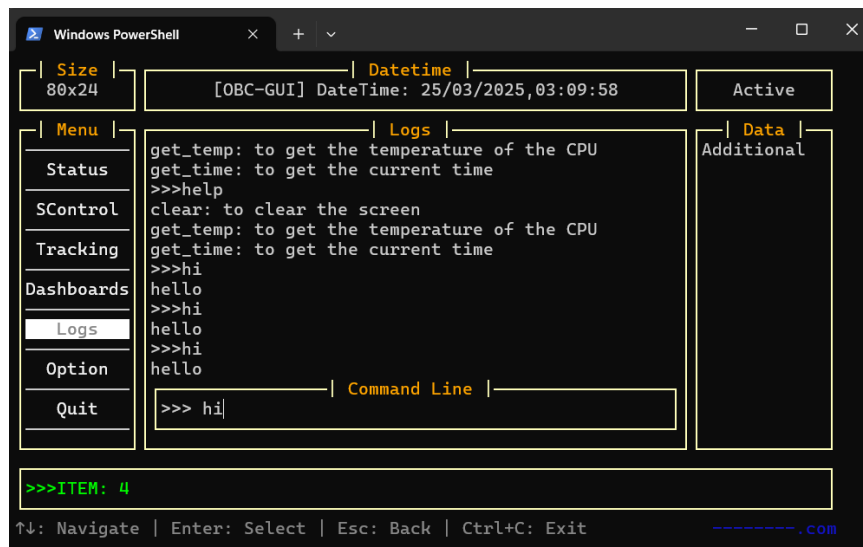
Quay lại màn hình hiển thị bảng, thì log đã cập nhật được tới cảm biến khác



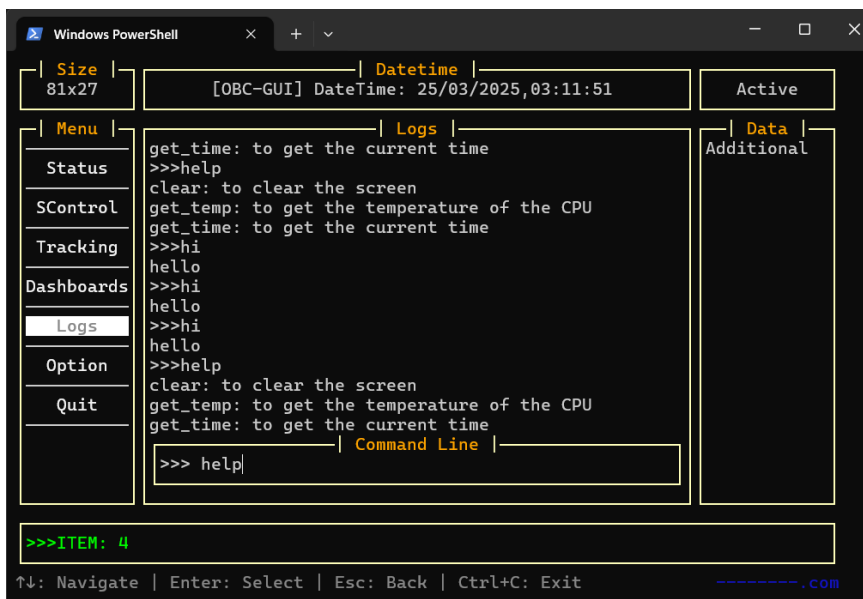
2/ Hiển thị lệnh khi nhập ở cửa sổ Logs



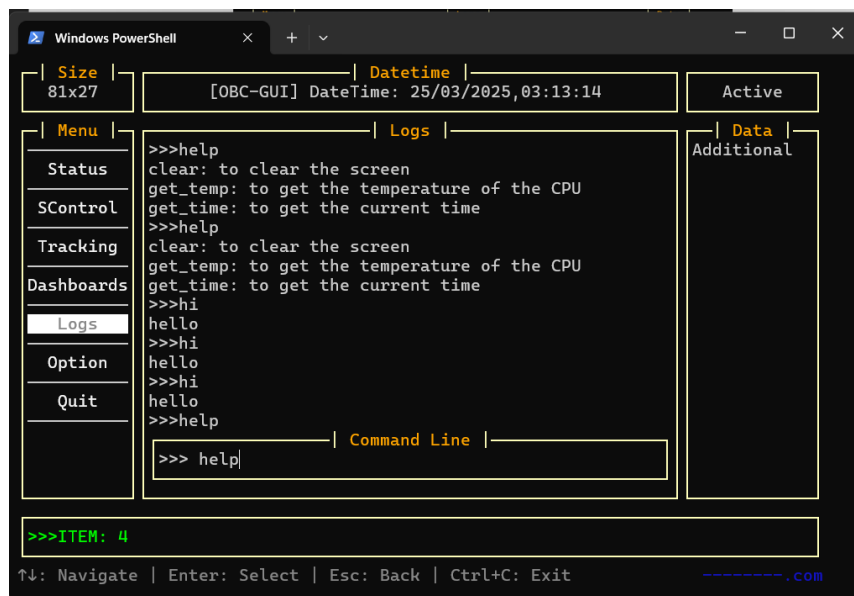
Cửa sổ sẽ tùy thuộc vào kích thước mà giới hạn khung dữ liệu hiển thị mới nhất



Khi thay đổi kích thước cửa sổ thì cửa sổ căn chỉnh lại sau khi nhập lệnh



Có thể dùng mũi tên lên để xem lịch sử cũ

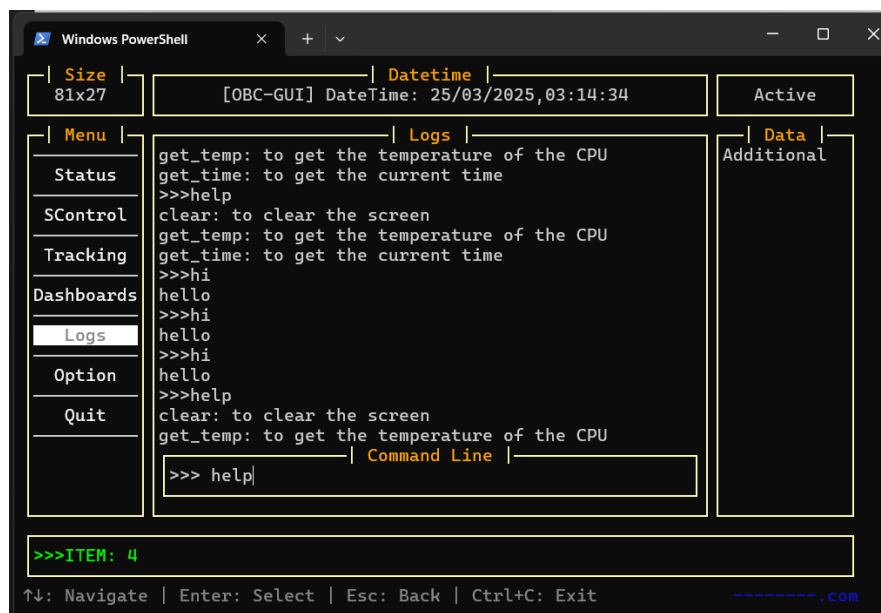


The screenshot shows a Windows PowerShell window with a terminal application. The terminal has a menu on the left with options: Status, SControl, Tracking, Dashboards, Logs (highlighted), Option, and Quit. The main area displays a command prompt with the following text:

```
>>>help
clear: to clear the screen
get_temp: to get the temperature of the CPU
get_time: to get the current time
>>>help
clear: to clear the screen
get_temp: to get the temperature of the CPU
get_time: to get the current time
>>>hi
hello
>>>hi
hello
>>>hi
hello
>>>hi
hello
>>>help
```

Below the command prompt is a "Command Line" input field containing the text ">>> help". At the bottom of the terminal, there is a status bar that reads ">>>ITEM: 4". The bottom of the PowerShell window shows navigation instructions: "↑↓: Navigate | Enter: Select | Esc: Back | Ctrl+C: Exit".

Dùng mũi tên xuống để xem lệnh(như là cuộn cửa sổ)



The screenshot shows a Windows PowerShell window with a terminal application. The terminal has a menu on the left with options: Status, SControl, Tracking, Dashboards, Logs (highlighted), Option, and Quit. The main area displays a command prompt with the following text:

```
get_temp: to get the temperature of the CPU
get_time: to get the current time
>>>help
clear: to clear the screen
get_temp: to get the temperature of the CPU
get_time: to get the current time
>>>hi
hello
>>>hi
hello
>>>hi
hello
>>>hi
hello
>>>help
clear: to clear the screen
get_temp: to get the temperature of the CPU
```

Below the command prompt is a "Command Line" input field containing the text ">>> help". At the bottom of the terminal, there is a status bar that reads ">>>ITEM: 4". The bottom of the PowerShell window shows navigation instructions: "↑↓: Navigate | Enter: Select | Esc: Back | Ctrl+C: Exit".

Lệnh clear dùng xóa màn hình

```
Windows PowerShell
| Size | | Datetime | | Active |
| 81x27 | | [OBC-GUI] DateTime: 25/03/2025,03:16:32 | | Active |

| Menu | | Logs | | Data |
| Status | | | | Additional |
| SControl | | | | |
| Tracking | | | | |
| Dashboards | | | | |
| Logs | | | | |
| Option | | | | |
| Quit | | | | |

| Command Line |
| >>> clear |

>>>ITEM: 4

↑↓: Navigate | Enter: Select | Esc: Back | Ctrl+C: Exit -----,com
```