**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 ấn 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 regrex 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 đinh vị trí.

- Nếu ấn mũi tên lên thì nó sẽ nhận event phím và thay đổi cái vị trí bắt đầu và kế thúc dựa vào kích thước terrminal 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 xủ lý khi đọc từ file.log về      def process\_data(self,log\_line):          patern = r"C(\d+)-(\d+)"\                  r".\*?**\[**T: 0**\]**-**\[**ADC: (\d+)**\]**"\                  r"(?:.\*?**\[**T: 1**\]**-**\[**ADC: (\d+)**\]**)?"\                  r"(?:.\*?**\[**T: 2**\]**-**\[**ADC: (\d+)**\]**)?"\                  r"(?:.\*?**\[**T: 3**\]**-**\[**ADC: (\d+)**\]**)?"          match = re.search(patern, log\_line)  #patern ở đây là 1 regrex 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, côt, 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)

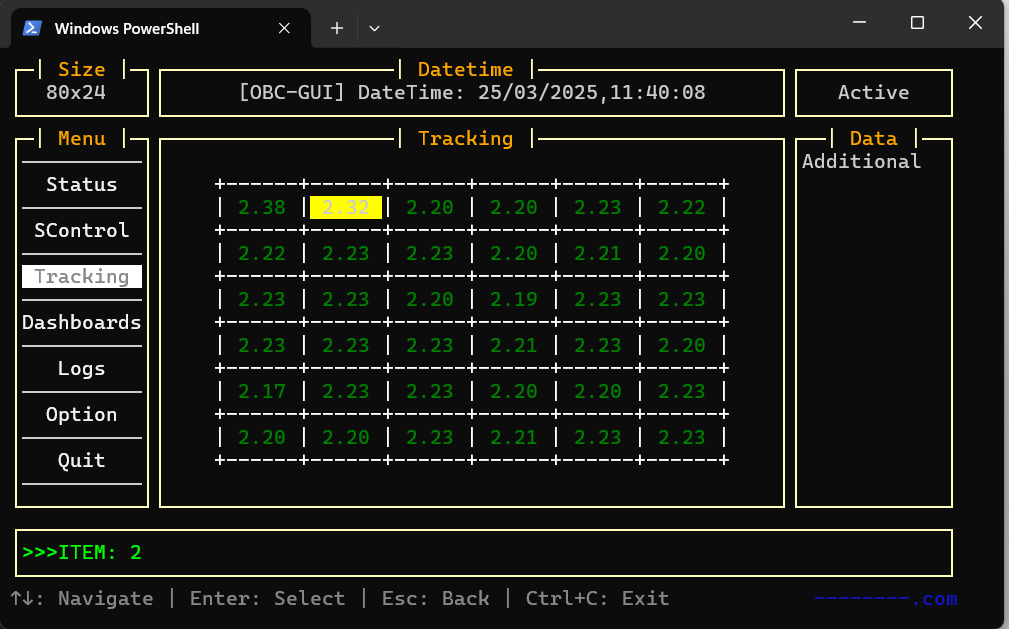
|  |
| --- |
| def setup\_keybindings(self):          #viết một hàm với sự kiện là nó sẽ thực hiện nếu ta bấm phím mũi tên lênlên          @self.kb.add('up')          def \_(event):                if self.mode == 'menu':                  selectable\_count = sum(1 for item in self.menu\_items if not self.is\_divider(item))                  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 sổsổ                          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 thôithôi                                  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 cửacửa                              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

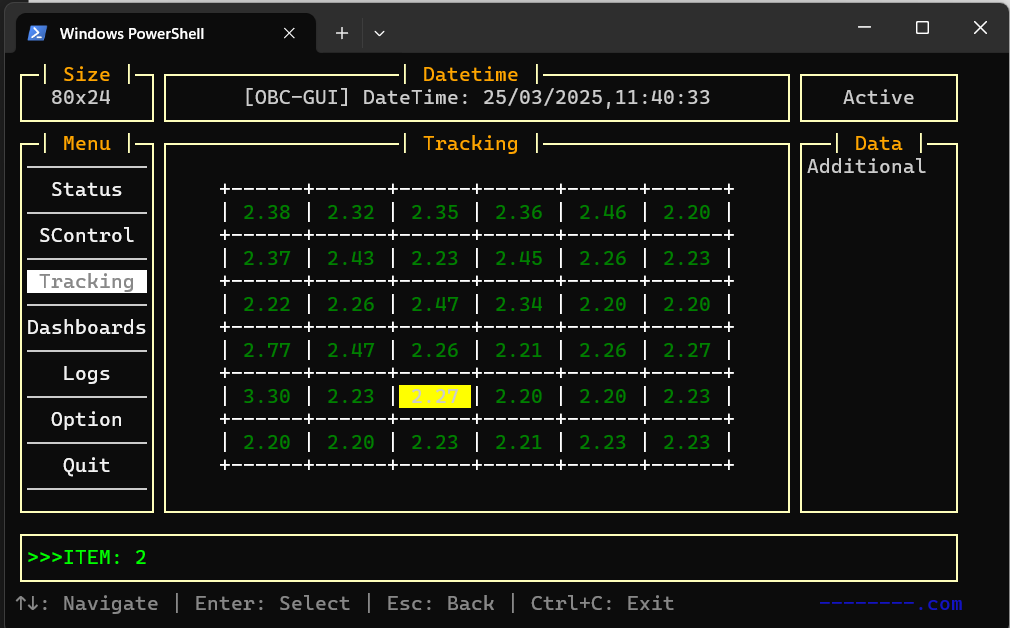
1/ Cửa sổ bảng hiển thị giá trị cảm biến



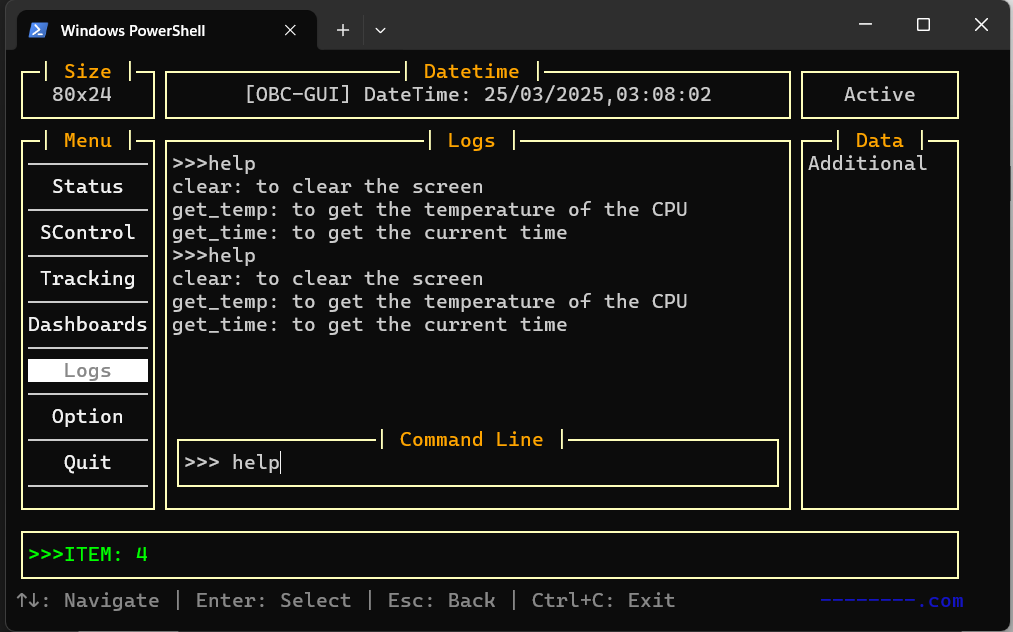
Thử chuyển sang cửa sổ khác



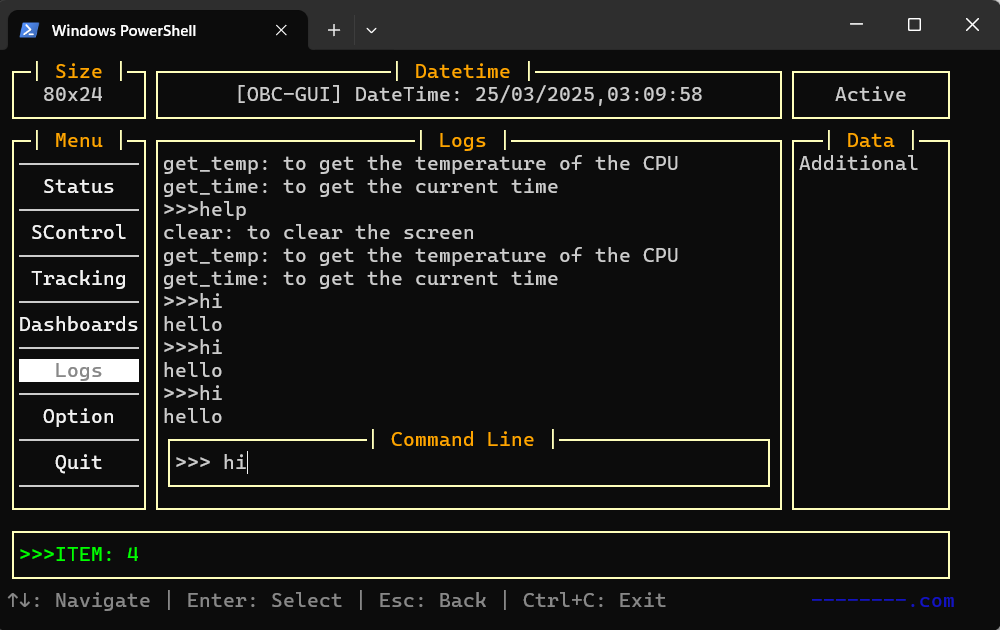
Quay lại màn hình hiển thị bảng, thì log đã cập nhật được tới cảm biến khác



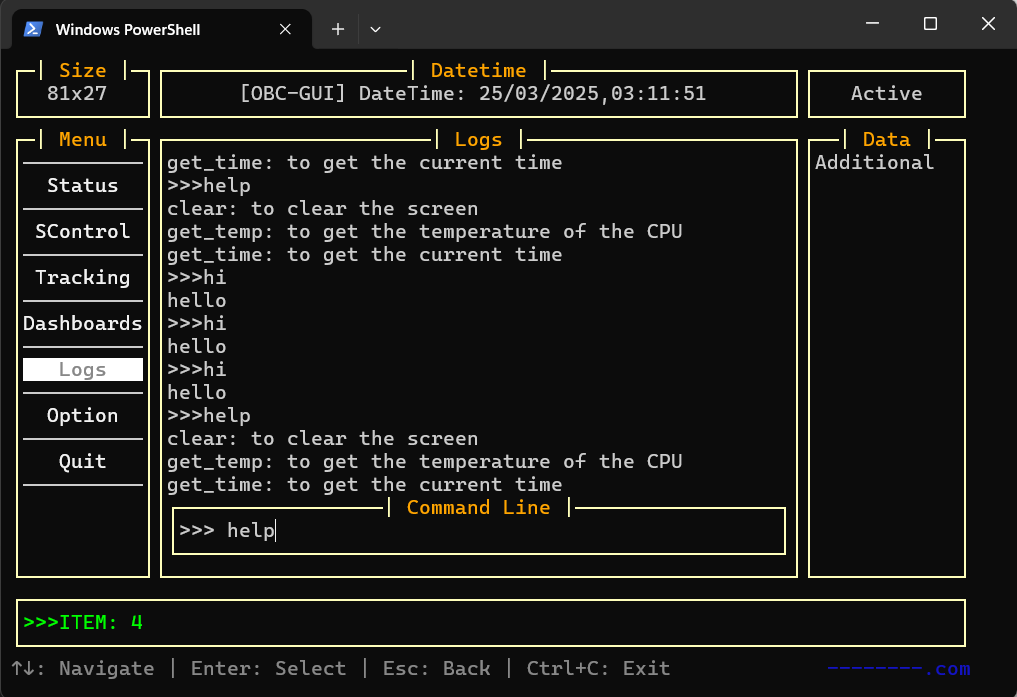
2/ Hiển thị lệnh khi nhập ở cửa sổ Logs



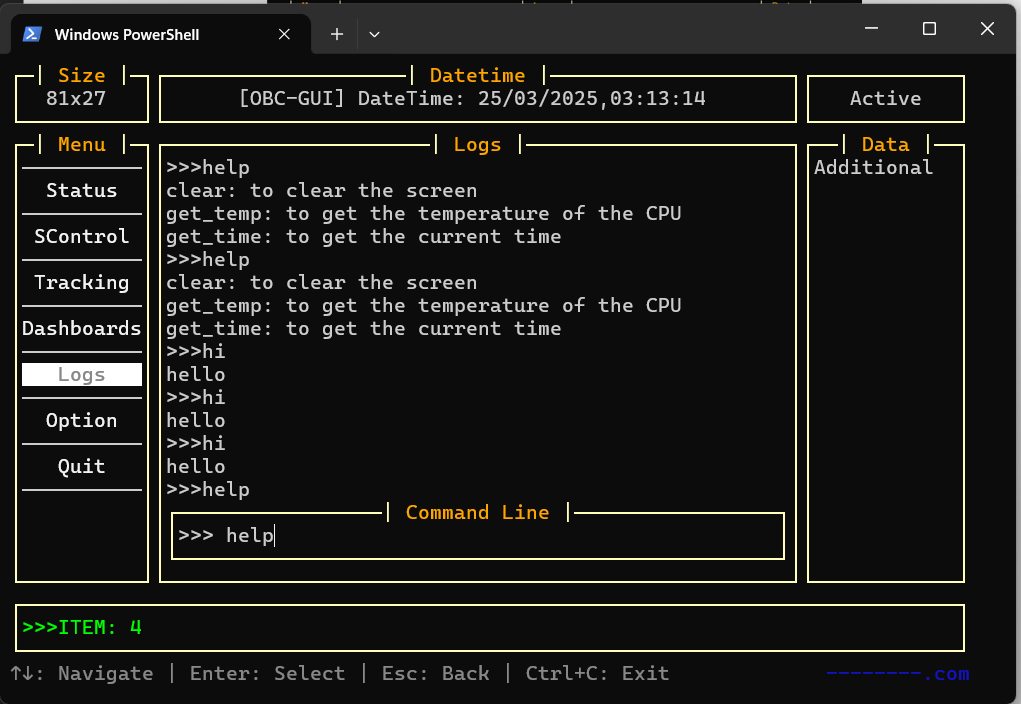
Cửa sổ sẽ tùy thuộc vào kích thước mà giới hạn khung dữ liệu hiển thị mới nhất



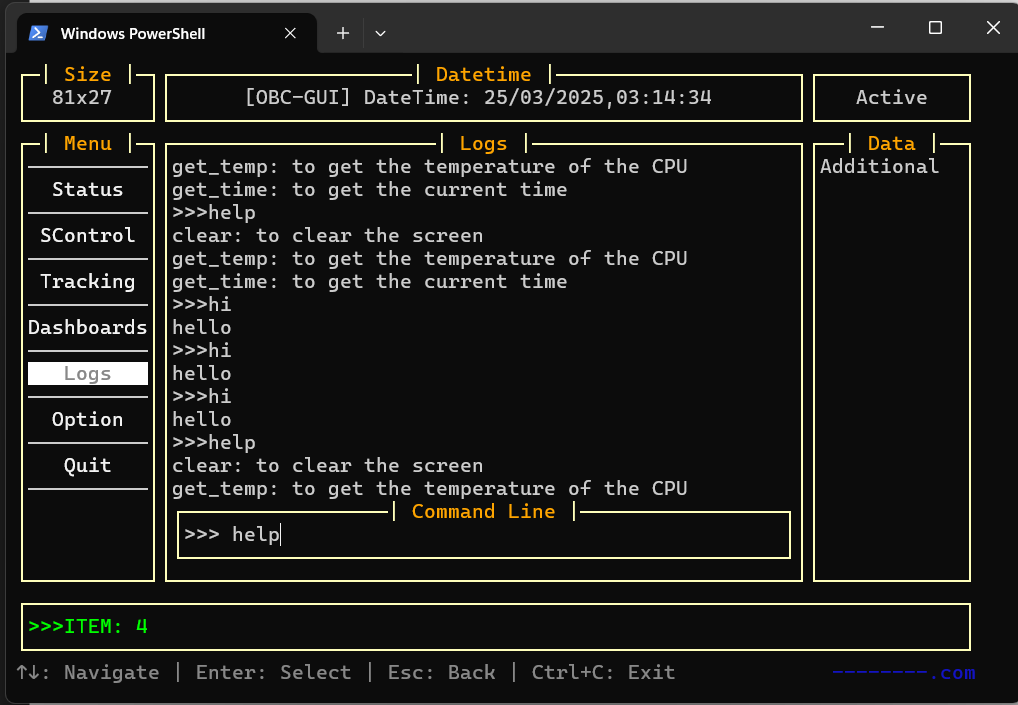
Khi thay đổi kích thước cửa sổ thì cửa sổ căn chỉnh lại sau khi nhập lệnh



Có thể dùng mũi tên lên để xem lịch sử cũ



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



Lệnh clear dùng xóa màn hình

