Handling Terminal Input/Output with Threaded Python

Brian Eng
2022-04

Handling Terminal Input/Output with Threaded Python

In a previous memo [1] I detailed the addition of concurrent execution to program that controls the PID process for controlling flow the use of threads to provide parallelism was introduced; in this memo I mention the development of that code to better handle the terminal interactions of those threads. Figure 1 shows the flow control for the threads.

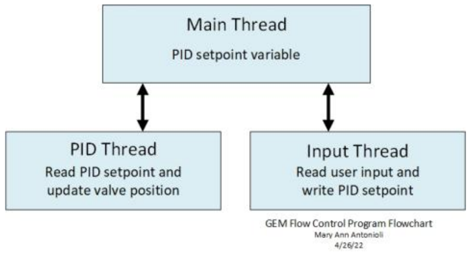
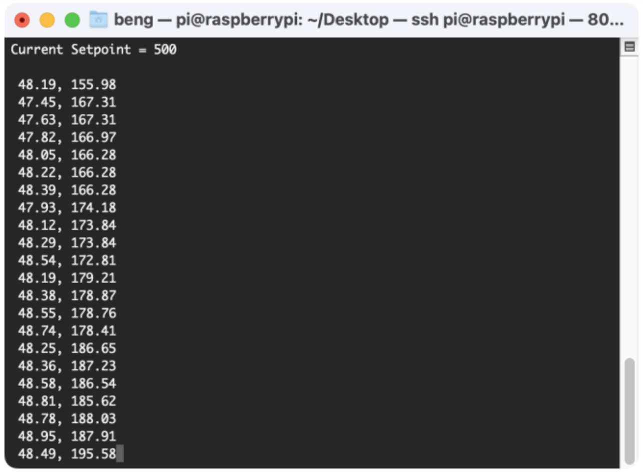

Main Thread
PID setpoint variable
PID Thread
Read PID setpoint and update valve position
Input Thread
Read user input and write PID setpoint
GEM Flow Control Program Flowchart
Mary Ann Antonioli
4/26/22

FIG.1. Thread Flowchart

When running multiple threads I found that while troubleshooting and adding additional console output logging, it was very messy and difficult to keep track of the source of the messages, especially when trying higher sample rates for the PID thread. A module that calls the curses library was therefore added to the program. The curses library provides terminal control via an abstraction layer for screen-painting and keyboard-handling. It also handles windows, a rectangular area on the screen, these windows are not required to be the same size as the terminal window itself. As the user input thread generally doesn't have much I/O to perform I gave it a window only two rows tall at the top of the screen while the PID thread would get all the remaining area, an example output is shown in figure 2. The top of the window shows the currently entered setpoint, which will be updated when a user enters a new value. The bottom of the window displays the current valve position and flow value, which will scroll up as newer values are output at the bottom of the window.

- **Curses library added to threaded python PID control code**
  - **Separate threads have their own window section for easier troubleshooting and the ability to precisely control text positioning**

Handling Terminal Input/Output with Threaded Python


beng — pi@raspberrypi: ~/Desktop — ssh pi@raspberrypi — 80...

```
Current Setpoint = 500

 48.19, 155.98
 47.45, 167.31
 47.63, 167.31
 47.82, 166.97
 48.05, 166.28
 48.22, 166.28
 48.39, 166.28
 47.93, 174.18
 48.12, 173.84
 48.29, 173.84
 48.54, 172.81
 48.19, 179.21
 48.38, 178.87
 48.55, 178.76
 48.74, 178.41
 48.25, 186.65
 48.36, 187.23
 48.58, 186.54
 48.81, 185.62
 48.78, 188.03
 48.95, 187.91
 48.49, 195.58
```

FIG.2. Example console output

The main issue encountered was that the coordinate system used in curses is passed as (y, x) rather than the normal convention of having the x position listed first. While extensively documented it is uncommon enough that it caused a few unexpected display issues when first trying to implement the code.

With the addition of the curses library to allow each thread to have a separate window of the screen it allows easier troubleshooting, a nicer overall display, and the ability to precisely position elements should additional information need to be displayed.

[1] Brian Eng, *Threaded Python for Improved Debugging & Performance,* Software Memo 2022-03, 2022.