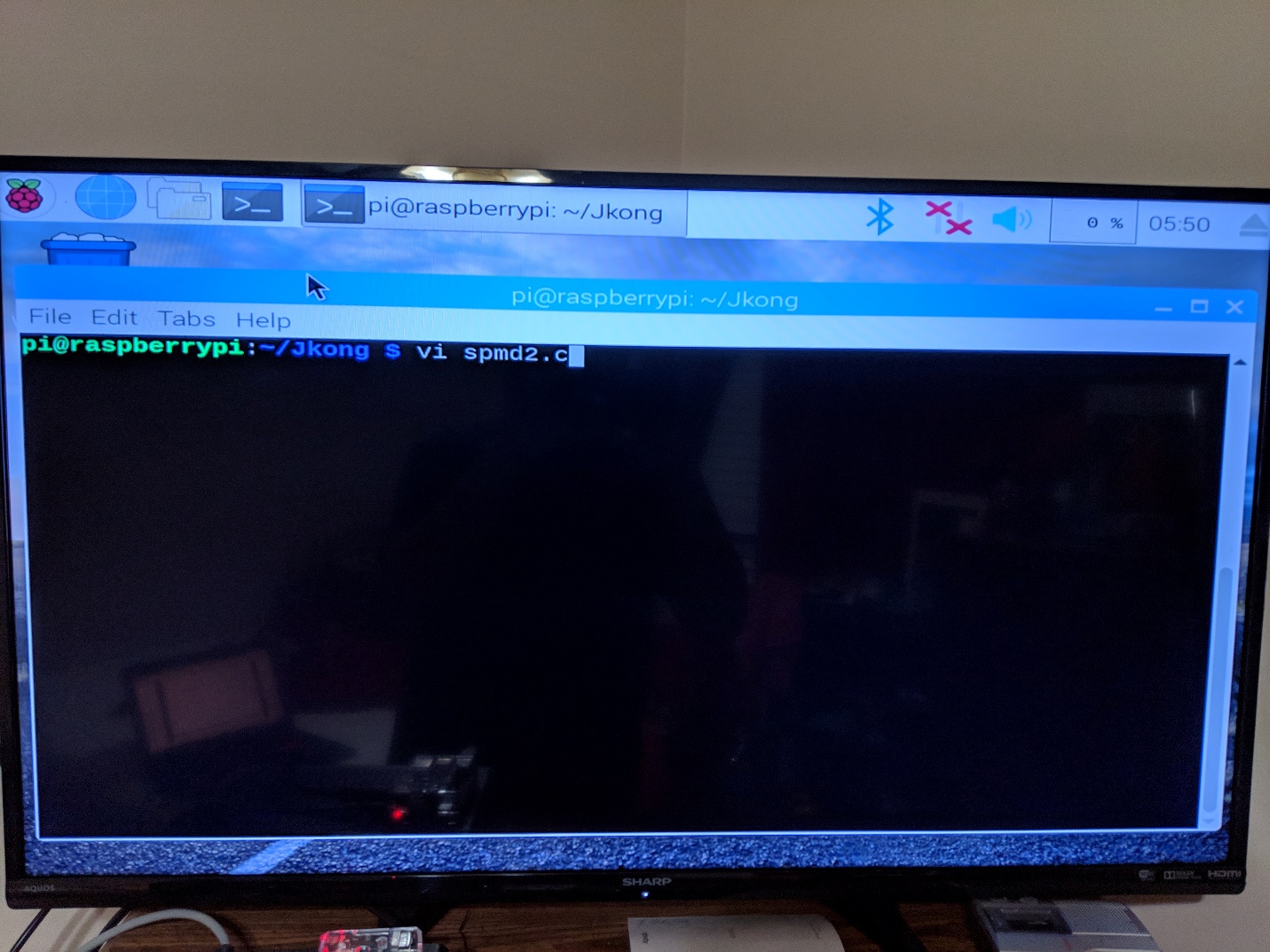
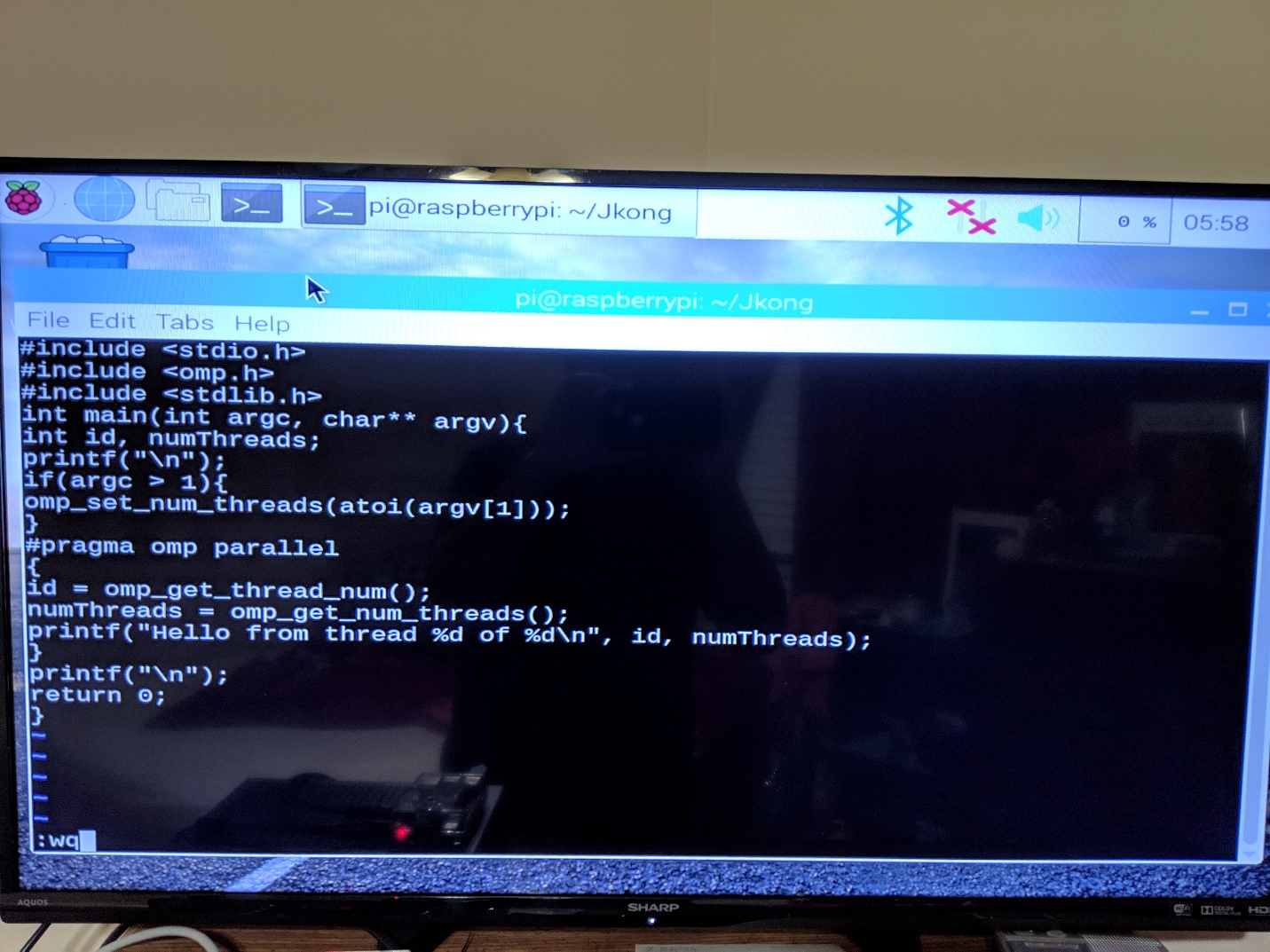
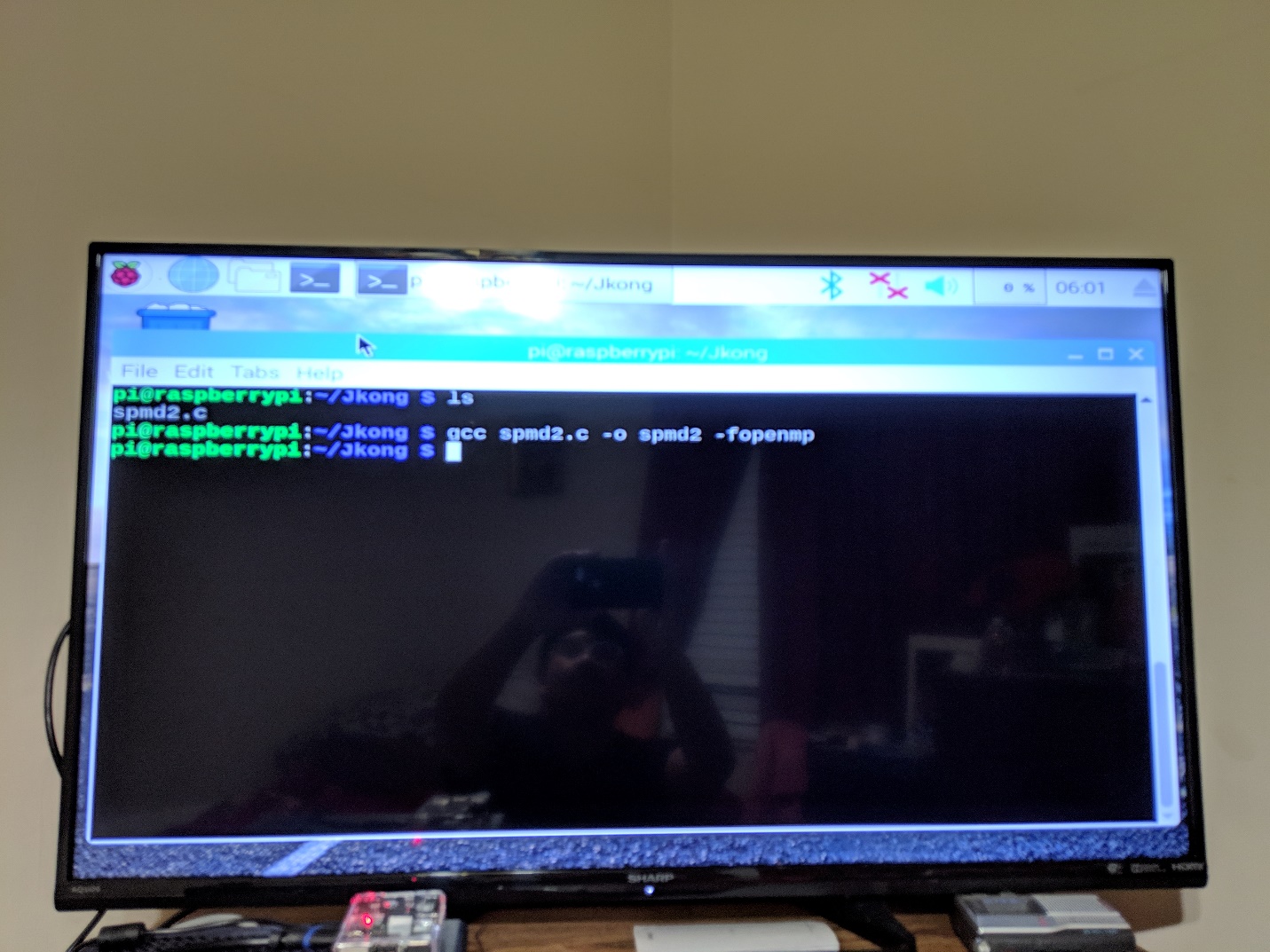
Task 3 Parallel Programming Basics:



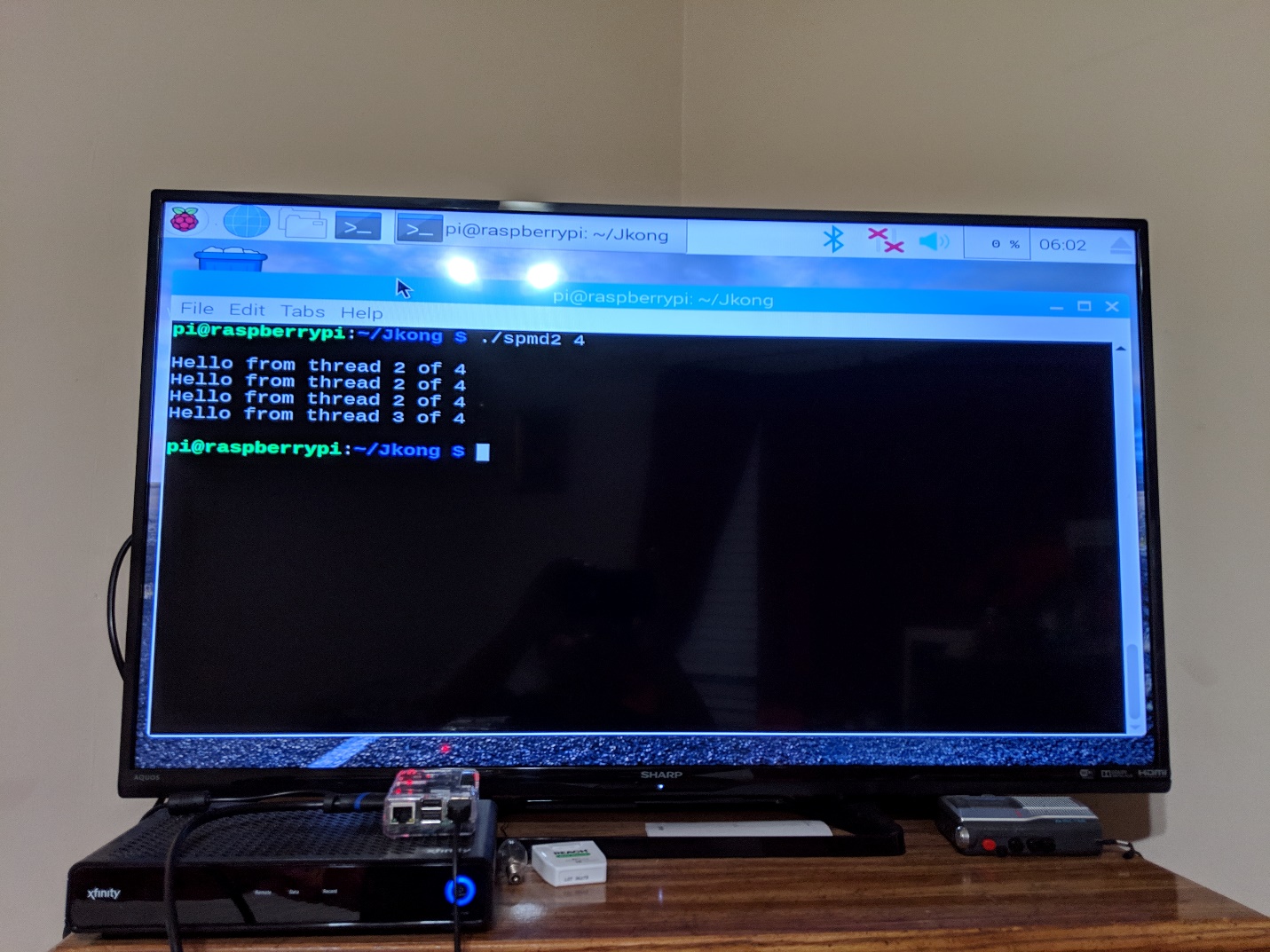
I have connected my Raspberry Pi to my monitor and booted it up. Having opened the terminal, I began by creating my own directory to organize the files inside the Raspberry Pi. The project task asks us to recreate spmd2.c file using the “nano” text editor however, I felt more comfortable using “vi” editor, so I used that instead.



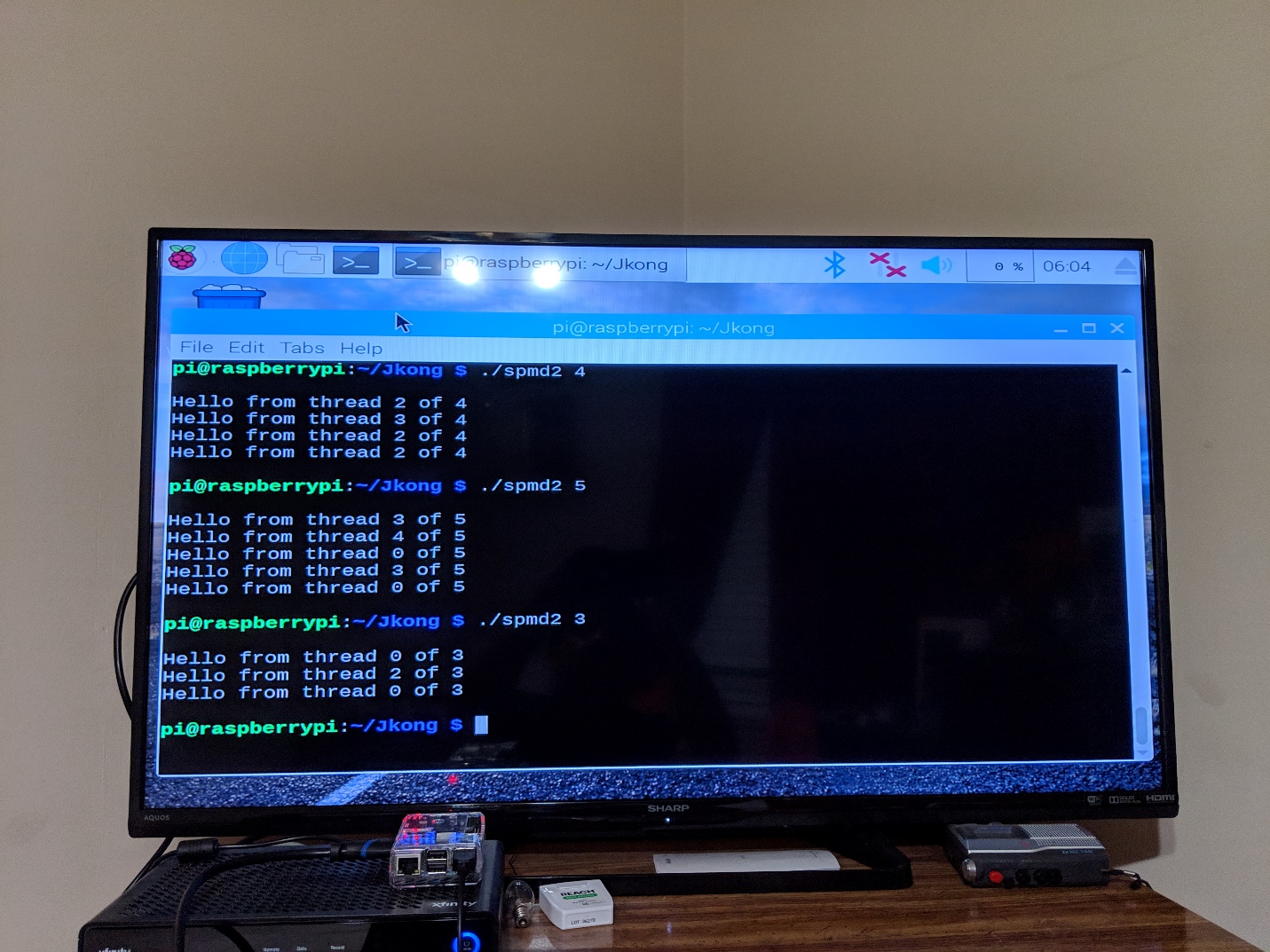
Here is my created spmd2.c file before I saved and quit the “vi” editor.



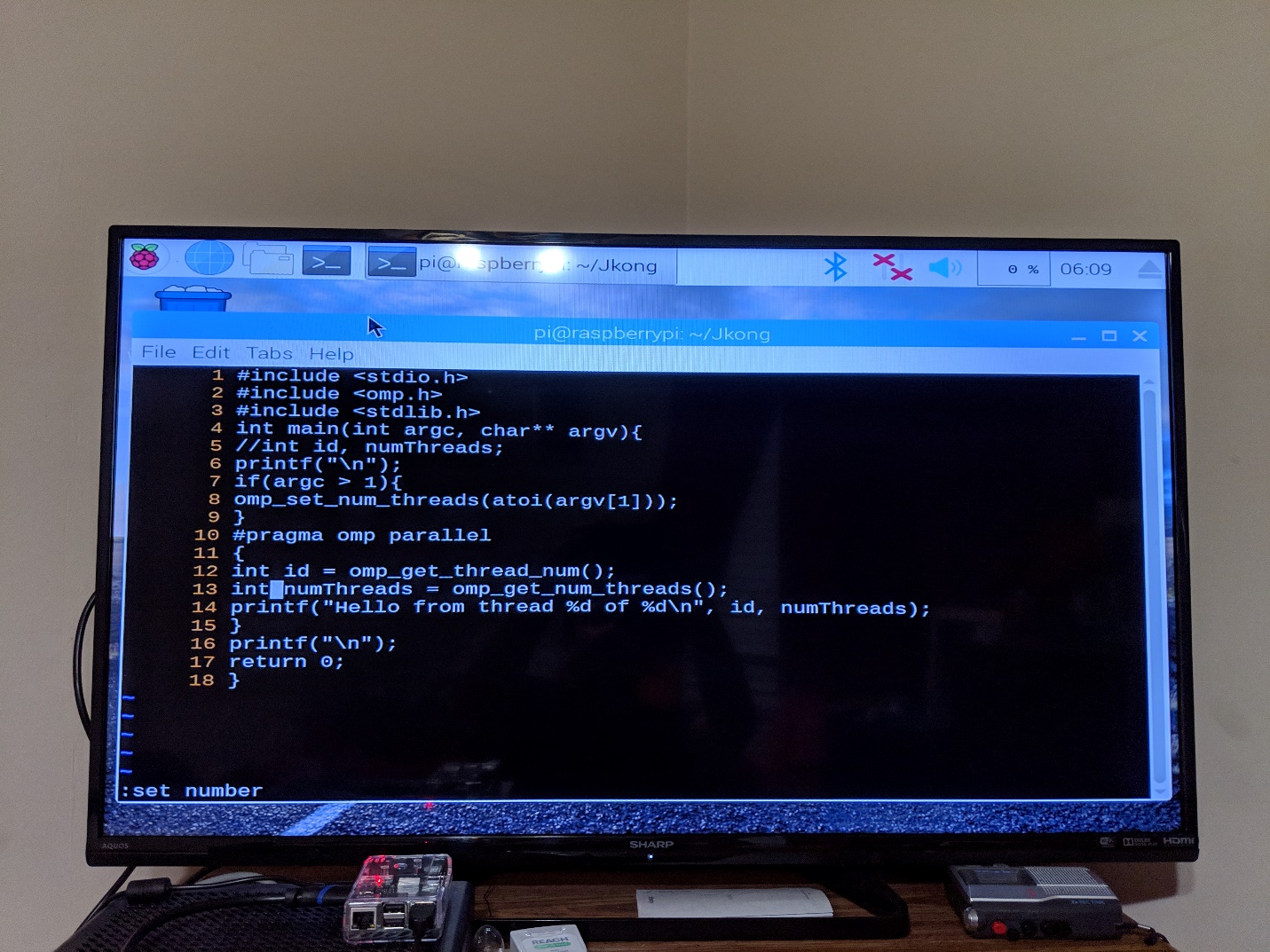
After quitting “vi”, I checked my directory to make sure the file is there. Then I compiled the file with the “gcc” command and used the “-o” option to rename the executable to “spmd2”. No errors were detected in the compiling of the spmd2.c file, so everything seemed good.



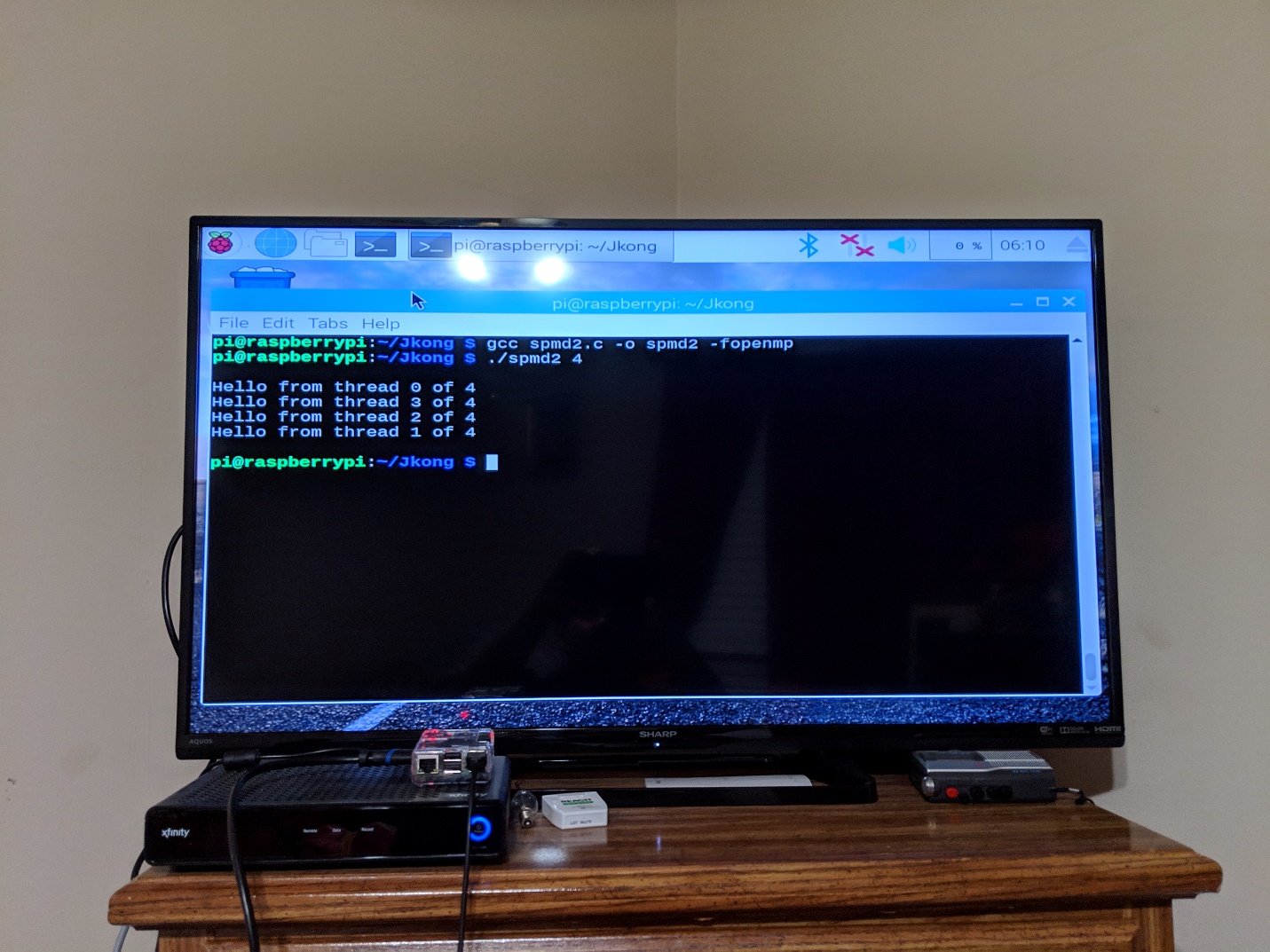
This is the output from running the spmd2 executable. At first, I was looking the output of each thread to be printed sequentially. For example: thread 1 will print then thread 2 will print and so on and so forth. But the programming task assured be that its normal since we couldn’t tell which thread will finish first. However, having the same thread id is not good. According to the programming task, improperly placing your variable declarations will cause your threads to share that variable’s memory space. So, I had to go fix it.



Here is me calling the smpd2 program with different number of threads before fixing it, just to see what would happen.



Back inside the “vi” editor, I commented out the initial declaration of “id” and “numThreads” that I wrote at the beginning of the programming assignment and initialized those variables inside the block of code that will be forked.



Here is me compiling the edited spmd2.c file and running it again. Each message has the proper thread id, which is what we wanted.