Final Project (Part 3)

CS 3410 Systems Programming

Due Date: Monday April 17, 2017 at 6:00 PM

You may work in teams of *up to* **three (3)** members from your lab section

Background Information

We are now going to make the system smarter by adding in knowledge about the user's surrounding environment. We will integrate an environment sensor (for example, a temperature sensor, humidity sensor, or barometric pressure sensor) into our system and correlate the readings with heart rate data. For now, we will not store the environment sensor readings in the histogram structure.

This component will also introduce a second process to the host program. One process will continue to run heart rate data collection and storage (as in Part 2) and the other will run a command line interface (as in Part 1). The goal is to allow data handling and process to happen in a background process and not interfere with the user-facing command line interface.

Specification (Part 3)

The command line interface will expose the following functions:

- `show X`: Set the output device to show `X` as the current heart rate instead of the current real-time value. In addition, print the value to the console.
- `pause`: Pause the output and keep the display device showing the current reading. Indicate this somehow in your circuit (using an LED for example).
- `resume`: Show the real-time heart rate on the display device. This should be the default mode of the system. Indicate this somehow in your circuit (using an LED for example).
- `rate`: Query the value of the heart rate sensor at the current time and print it to the console
- `env`: Query the value of the environment sensor from the Arduino and print it to the console.
- `hist`: Print a representation of the current time block's heart rate histogram to the console
- `hist X`: Print a representation of the given time block's heart rate histogram to the console
- `reset`: Clear all data from the backing file
- `exit`: Exits the host program

Deliverables and Grading

- Push your code to GitHub (a link will be posted on Blackboard) before the deadline. The following things should be in a directory named `part3`:
  - One Arduino sketch
  - The C source code (and appropriate header files) of your host program. Be sure to include a `Makefile` so the grader can build your code. Do not forget, your code *MUST* compile with the following `gcc` flags: `-std=c99 -Wall -pedantic -Werror`. You will lose credit if your code compiles, but only without those extra flags.
  - One Fritzing[1] file that shows the circuit you built.
  - A document explaining how you have extended your communication protocol to support the new features. Additionally, provide a complete, updated, protocol specification.

[1] http://fritzing.org/home/

- You will be giving a demo of your circuits before class on the day of the deadline. Be sure to bring everything you need to show it off.
- Be sure to write clear and concise commit messages outlining what has been done.
- Write clean and simple code, using comments to explain what is not intuitive. If the grader cannot understand your code, you will lose credit on the assignment.
- Draw clean Fritzing schematics. If the grade cannot understand them, you will lose credit as well.
- Be sure your code compiles! If your sketches do not compile, you will receive **no credit**. It is better to submit a working sketch that only does a subset of the requirements than a broken one that attempts to do them all.

Table 1: Grading Rubric

Category	Percentage
Demo	60%
Code Quality (including `Makefile`)	15%
Write-Up & Protocol Design	10%
Compilation with `-Wall -pedantic -Werror`	10%
Schematic	5%