# Final Project Part 4

Aaron Coplan, Samsara Counts, Seamus Malley

## Full Protocol Specification

Our communication protocol behaves in a manner where the C code acts as the *master* and the Arduino sketch acts as the *slave*. The C code sends data in the form of `c2ar_pkt`, which involves a single byte indicating a code to do one of the following actions:

- OUTLIER (0): The last value recieved was an outlier, light the LED to indicate this.
- VALID (1): The last value recieved was valid, turn the LED off to indiciate this.
- DATA_REQUEST (2): Request bpm and date data from the Arduino.
- SHOW (3): Show the value that will be sent next on the LCD.
- PAUSE (4): Pause the LCD screen on the current value.
- RESUME (5): Resume the LCD screen into real time mode.
- ENV (6): Request environment and date data from the Arduino.

The commands Outlier, Valid, Show, Pause, and Resume do not require reading a response from the Arduino, but the commands Data Request and Env do. This response is read in as a string formatted as `%d %d %d %d %d %d %d` ordered as bpm, day, month, year, seconds, minutes, hours. This data is then validated to ensure nothing is abnormal and all data has been read. At that point, it is put into a `ar2c_pkt`, which contains an `int` representing the value of the heart rate sensor or environment sensor, and a `struct tm*` to track the timestamp. The robustness of this protocol comes from the designated structure and error checking we employ.

## Database Schema

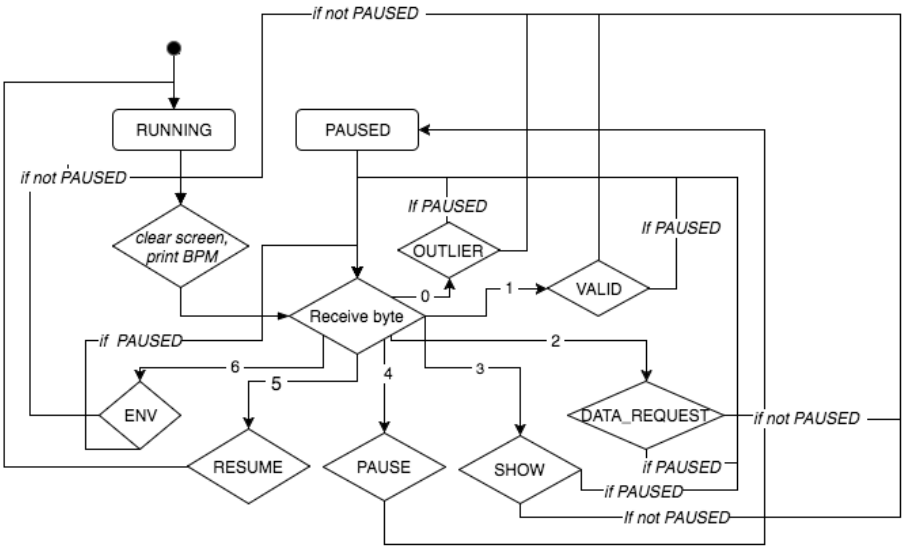
Our database schema is as follows:

```
BPM(minute_num INTEGER, bpm INTEGER)
ENV(minute_num INTEGER, env INTEGER)
```

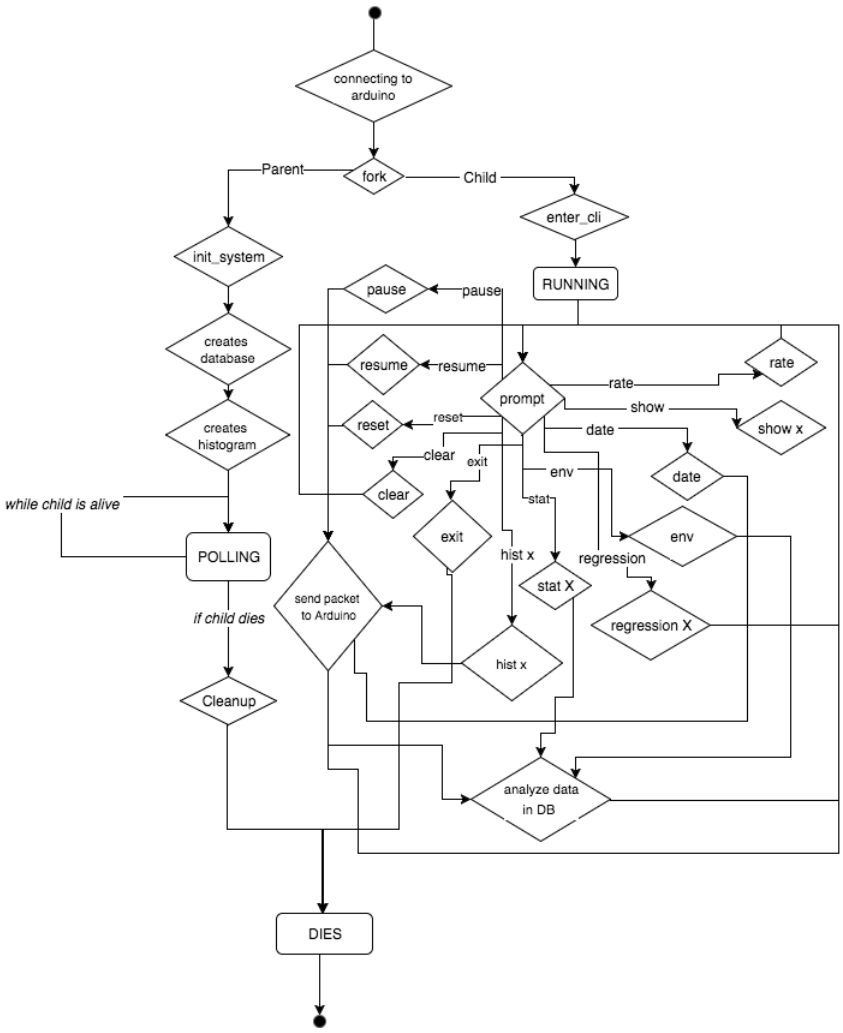
Thus, we are able to `GROUP BY` and `JOIN` using `minute_num`, which made building sensor reading pairs very simple (this is explained further below).

## State Machine Diagram

Arduino




C Program




## Additional Specifications

- Outlier Detection
  - To detect ouliers, we first wait until the program has collected at least 15 data points for the current bucket. Within the first 15 data points, a reading is only an outlier if the heart rate is outside the range 30-200. After the first 15 data points, points that are not within two standard deviations of the mean are considered outliers for the current bucket.
- Time Synchronization

  - Upon starting, the Arduino waits for the C code to send the time data. Once it has recieved the data over Serial, it forwards it to the RTC, setting the time. Once this exchange has taken place, the programs each enter their respective state machines.

- Building Sensor Reading Pairs
  - For regression analysis, we build sensor reading pairs using SQL. The SQL query `SELECT bpm_avg.avg_bpm AS bpm_reading, env_avg.avg_env AS env_reading FRO` successfully pairs readings by minute numbers within a particular bucket. We then use these pairs to calculate the regression.

- Avoiding Concurrency Issues
  - To avoid having concurrency issues between the CLI process and the data polling process, we have the data polling process store its last bpm and env reading into mmap'd memory. When the `rate` or `env` command are called from the command line, it simply accesses the corresponding value in that mmap'd memory instead of having to make a request to the Arduino, as that may cause concurrency issues.