***Database schema Proposal:***

**Overview:**

 For the cube works project we are collecting and storing data at a regular interval. As such we must implement a database structure. We also have the problem of transmitting the data after. Due to the sensitive nature of the transmission time and trying to find the easiest and most efficient way of database implication, I believe that a two-part system would be best for the cube works mission.

Part one: The main thread,

             This thread handles the receiving of the transmission command from ground, creating transmission files, sending the transmission mode command to the watchdog, and all other normal operations of cube works.

Part two: The transmission thread,

             This thread handles the loading of the radio buffer, verifying that the transmission files are correct, letting the watchdog know when it is ready to be reboot (if applicable), and giving the good or bad transmission flag.

The watchdog:

             This is an Adruino beetle that is wired into the pi that checks to see if the battery voltage is sufficient to run the pi. It also checks to see if the pi is still running and responsive.

**Proposal:**

The database: For ease of use and efficiency it is proposed that we use sqlite3 as our database structure. This is a prebuilt database schema that is compatible with python, it is very compact, and provides an easy to use platform with support.

Transmission Protocol: We have about 500 milliseconds to load 128 bytes into the radio’s buff before it sends, or we miss our window of transmission. Due to the nature of python, and the sort pried in which we must transmit. It is proposed that we have a multi-thread/async operation to handle a transmission. (Due to the nature of the raspberry pi it may not be possible to use multi-threading, in this case, we would use async. See *notes pg 3*) The logic of said operation will go as follows.

1. We receive the transmission from the ground station.
   * 1. The transmission specifies the time interval in which to send data back to the ground station, the interval to wait before transmitting, and the data requested.
2. The data is prepared.
   * 1. The pi will query the database and gather all the information need into corresponding files.
3. The “main” thread will send the transmission code to the watchdog and the time in which it needs to transmit, the watchdog will keep this in mind in case a system reboot is needed it can plan accordingly. See: *Database and watchdog, pg 2.* The “main” thread will wait to receive the code telling it if a reboot is necessary.
4. It will then trigger the second thread, passing it the transmission interval, delay interval, and if a reboot is needed or not.
   * 1. The “main” thread will then return to normal flight logic
5. The transmission thread
   * 1. The transmission thread will determine the order in which to send the requested data.
     2. It will then open the files in that order to make sure they are functional.
        1. If a file should fail to open, the transmission thread will raise a flag telling the main thread to recreate that file.
     3. If a reboot is needed it sends the ready code to the beetle.
     4. The transmission thread will then wait what remains of the requested wait interval.
     5. The transmission thread will send the data to the radio.
     6. Transmission thread closes. If the data transmission was good, then it closes with the “Good” flag. Else it closes with the “Failed” flag. (The tread will close after the said interval of transmission.)

Note: the transmission thread will be written using C. (This will ensure maximum speed.)

1. The main thread evaluates the data transfer.
   * 1. Should it receive the data transmission good flag, the main tread will delete any info “no longer needed”. This info is TBD.
     2. If the main thread receives the Failed flag it will delete data that is no longer relevant ie, latest status, and corresponding data. It will then check the files, boot history, radio status ect to try and determine the cause of failure to transmit, a record that, and prepare it to be sent on the next transmission.

DataBase and Watchdog:

Logic flow:

1. The beetle receives the state transmit code, the interval of transmission, and the interval of delay from the pi.
2. The beetle checks battery current voltage and determines if there is enough power to last and transmit at said time.
3. If there isn’t enough power it will send the reboot need command to the pi and then wait for the transmission thread to say it has completed the file check.
   * 1. If the beetle receives the ready code it then checks the battery voltage to see if a reboot is still needed. It also checks how much time is left until the transmission interval.
     2. If the pi has enough time to turn off and then reboot. Then the power is cut and the beetle waits until there is about “10 minutes” (this interval is TBD, but will probably be on the order of 10 minutes), until the data transmission to turn on the pi.
     3. In the case where the watchdog decides it needs to reboot the pi, but the pi never responds with the ready code. The beetle will wait until the delay and transmission intervals have expired, and then it will reboot the pi.
4. The beetle will then return to its normal operation state.

***Pro/Cons***

*Pros*: T

* This system will be easy to build the database.
* It will be very fast on the transmission side of things.
* It has built-in power checks.
* By multi-threading it, we can help ensure that the “normal operations” of the pi do not interfere with the transmission.

*Cons:*

* + - Complexity,
      1. It will be more complicated to program a multi-threaded approach
      2. It will also require a lot of communication between the pi and the beetle
    - When using multiple files, the c code will be slightly slower as it will have to open said files

***Final Notes:***

* + 1. It should be noted that we are expecting a delay on the order of a couple of hours between when ground ask to transmit, and when we send data. This makes it so that slower processes like opening/creating files are negated.
    2. It may be more efficient to have the “main” thread tell the watchdog it is transmitting as soon as it receives the command from ground. To ensure that a reboot does not happen during file preparation.
    3. It is possible to do this proposal without the telling the watchdog however this could lead to two possible failures,
       1. One, low power failure. (Although we can check for this on the pi, it makes little sense to do so, because the pi can tell the beetle to turn it off, but it cannot tell it to turn it back on.)
       2. Two, the beetle reboots the pi during transmission. This would lead to transmission failure.
    4. It is possible to immediately send the reason why transition failed to ground; however, this code may not be received by ground.
    5. The pi may not be capable of multi-Threading which means we would need to use async. This probably will not be as “safe” as multi-threading. However, it will still protect the transmission process.
    6. It is important that the beetle checks to see if there is enough time to reboot or the beetle will reboot the pi during the transmission interval, and we may lose time to transmit or completely miss the window.
    7. Items that have “” around them. This means that we have not yet defined exactly what these items will do and how they will be defined.

***Conclusion:***

             It is the author’s *opinion*that this will be the best form of handling transmission. For the following reason:

1. This will protect the transmission tread.
2. This will ensure greater speeds in transmission.
3. It will be provided power checks to ensure transmission is possible.
4. It is possible to “pre-load” data to increase speeds even more.

The author is also of the opinion that the Cons due to complications are valid. However, the main objective of this mission is to receive a picture from the cube sat. Therefore, the transmission prosses should be where the most/best work is placed. If all other operations fail, then this proposal should provide safety for our transmission operation. In theory, this should be the last operation to fail. (Note: This does not include hardware failure.) The author believes that although this will require more work, it will be worth the effort, as it insures a greater chance of mission success.

Respectfully Submitted: Shawn Jones

Programming team lead, GAS team, USU