Going to have 2 tables

* One will hold the current state information
* The other will be a logger for state changes and updates to the system over time

State

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| BUSY/IN TRANSIT | Last floor position | Door open - close | Time last updated | Updated By |  |
| 1-0 | 1-3 | 1-0 | timestamp | Webserver or supervisor |  |

* BUSY/IN TRANSIT
  + Will be a the flag to state if the cart in current acting on a task
* Last floor position
  + This will be updated as the elevator passes a floor (checking busy will let you know if it is stationary)
* Door
  + Flag for door open vs. close logic
* Timestamp
  + Whenever anyone writes to this this is updated automatically
* Updated by
  + Lets us know who made the state change;
  + local hardware(SV knows there is a state changed because it had to do it) vs webserver(SV must update and send out the proper message for the change)

Logger

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Can Message | Can sender | Can receiver | Raw CAN | timestamp | Local access flag | Message type |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

* Can message
  + English style message briefly describing the log
  + example :
    - floor 1 requested the elevator
    - Elevator has passed floor 2
    - Elevator on rout to Floor 2 with care door open
* CAN sender
  + The address of the sender
    - Floor 1-3
    - SV
    - EC
* CAN receiver
* Raw CAN
  + The raw CAN message for debug/extra info
* Timstamp
* Local access
  + the webserver does not act as a real node with a different address it simulates being one. Because of this it would be nice to know if a can message was originally sent from the web server (pressing floor 1 in the car vs on the website should be different in some why for debugging )
* Message types
  + Updates (messages that are generated based on request- passing by a floor)
  + Error message (can messages that make it through the filter but failed when parsed)
    - Debugging purposes
  + Request (messages that are requesting a node to act/do something)

## Database table types

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| BUSY/IN TRANSIT | Last floor position | Door open - close | Time last updated | Updated By |
| char | char | char | Datetime | char |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Can Message | Can sender | Can receiver | Raw CAN | timestamp | Local access flag | Message type |
| varchar(50) | varchar(10) | varchar(10) | char | Datetime | char | char |

Python library - python-mysqldb

New SV logic will have it pulling the logger for messages that aren't local and act upon them. After the sv has forwarded the message along to the CAN the state table will then update with the new current information.

Only the sv modifies the state table entire

But both the sv and the web server update the logger with new entries