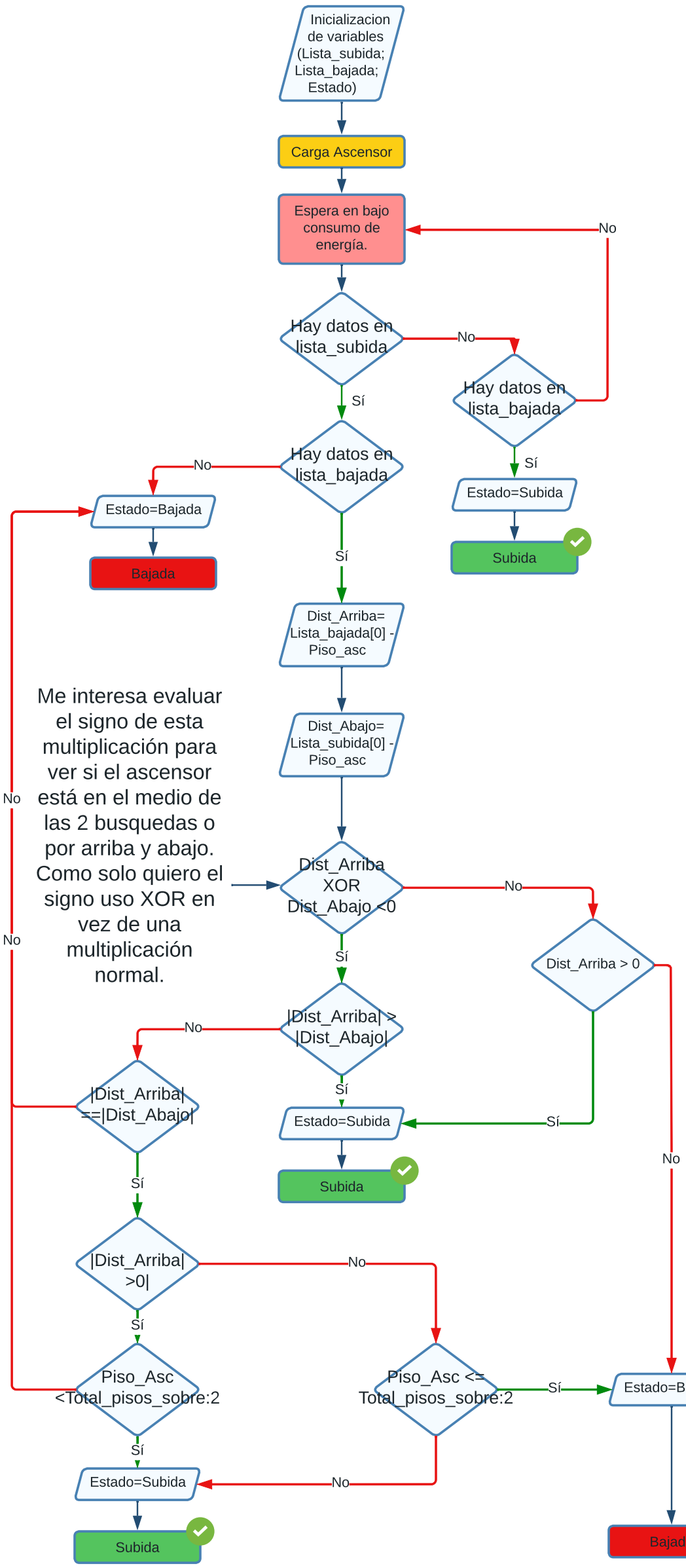
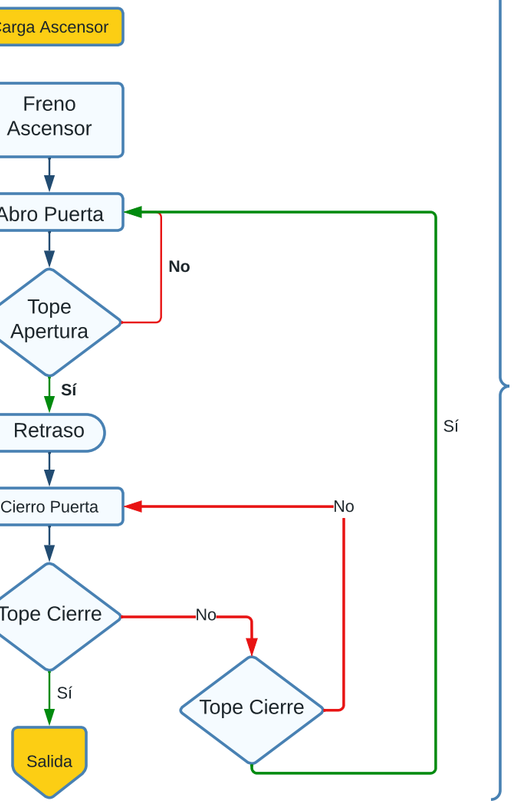


Inicialización del ascensor asegurandose de que siempre empiece en el punto que queremos.

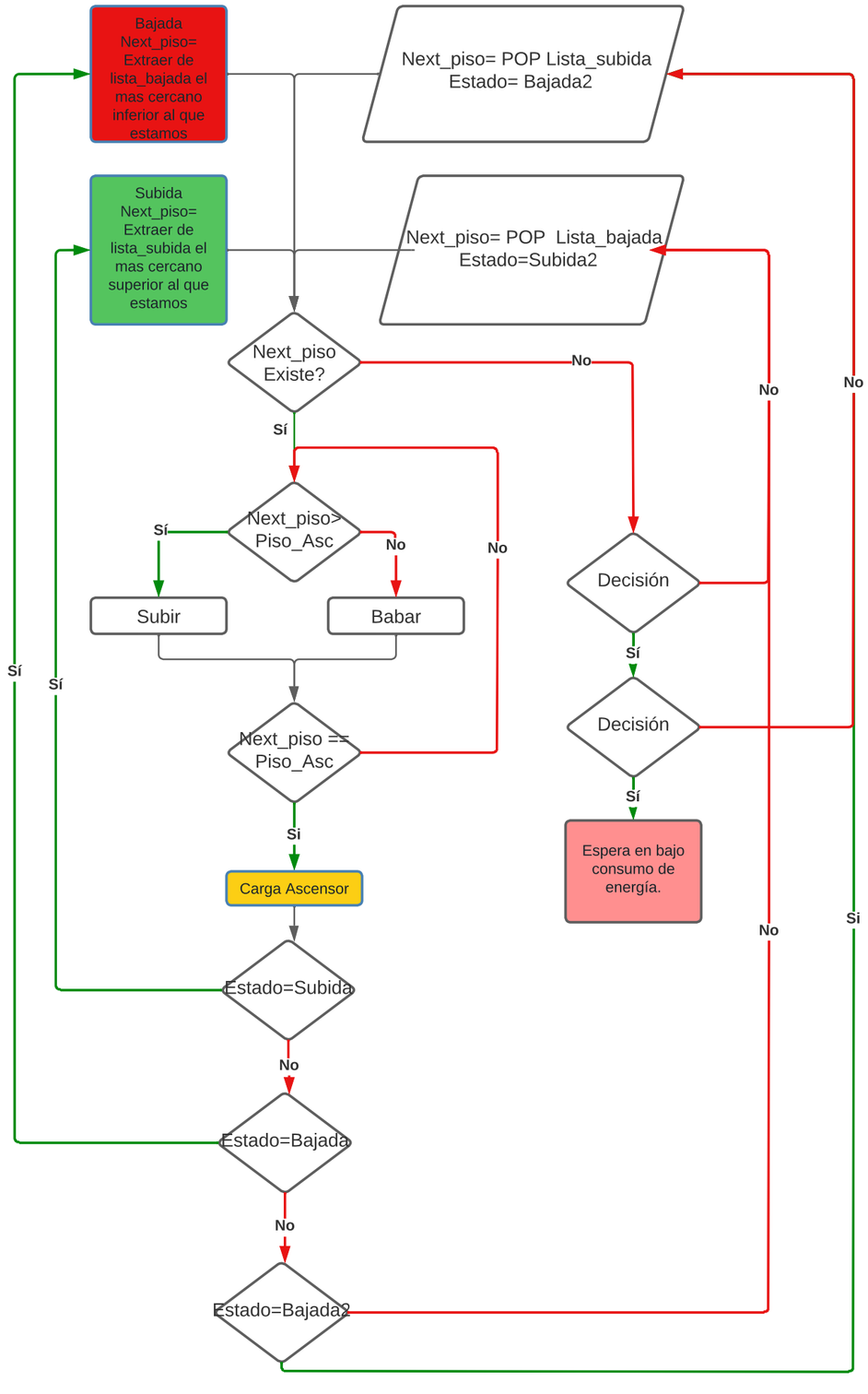
Analiza la existencia de datos en lista_subida y lista_bajada y eligé que camino tomar Optimizando los tiempos del recorrido basado en la combinatoria de los 24 casos que se me ocurrieron que pueden haber.

Subida: Si solo hay gente que quiere subir, voy a buscarlo. Bajada es el nombre que le puse pero contempla también el caso de que en vez de bajar a buscarlo tenga que subir a buscarlo

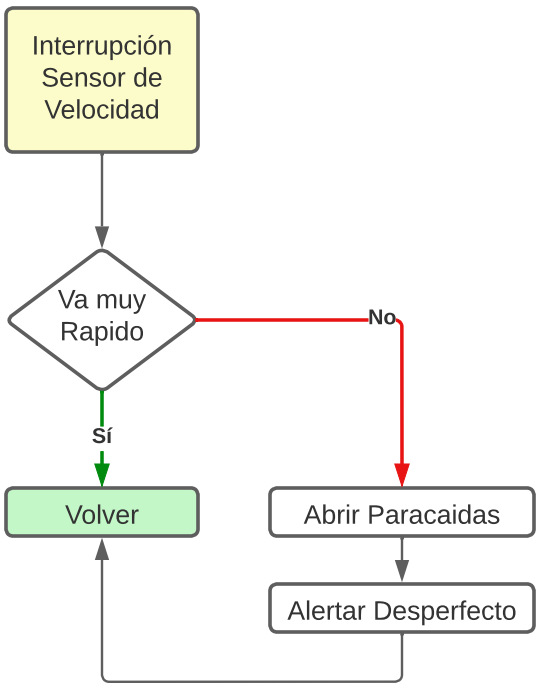
Bajada: Igual que Bajada, solo es el nombre que usé para decir. ir a buscar al que quiere bajar. Esto lo puse porque generalmente el ascensor subirá a buscarlo pero también podría bajar a buscarlo.



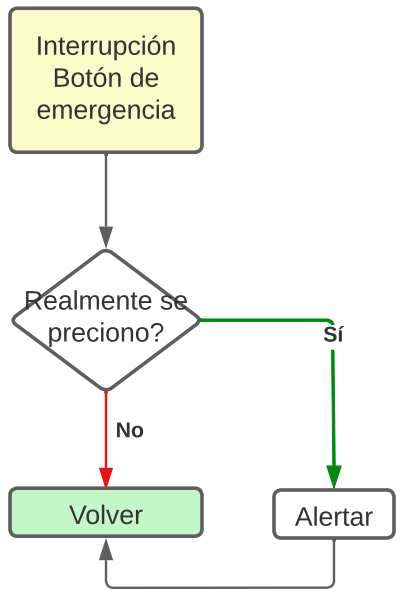
Control de carga para la seguridad en el ascensor y que nadie se quede atrapado por la puerta.



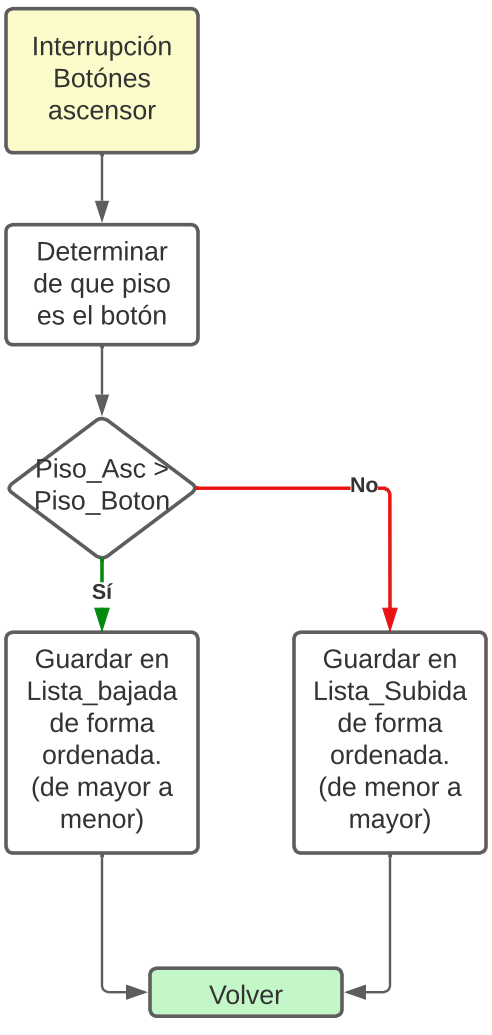
Son dos funciones que por su semejanza se pusieron en un mismo algoritmo identificadas por estados. La función SUBIDA sube llevando usuarios para los pisos de arriba pero sin bajar a buscar a los de abajo y una vez que llega al ultimo usuario "sin mirar atrás" va a buscar al usuario de más arriba y comienza a ejecutar toda la lista_bajada hasta terminar. La función BAJADA hace lo inverso. Ambas son capaz de subir y bajar por lo que al poner el comparador con las 2 funciones no estoy agregando código de más. A su vez como cada una de estas funciones tiene una parte de hacer la tarea opuesta dentro de sí se generá otra rama por cada una pero como ya estaba utilizando estados solo extendi a dos estados más (SUBIDA1 y BAJADA1) y utilicé la misma porción de código. Lo que llamé estado acá no es el estado en un smart contract sino una variable que marque el estado de ejecución de mi programa. Una vez que termina de ejecutar la función llamada lo volvemos a estado de bajo consumo de energía para esperar la próxima.



Usar Lógica negativa para encender el paracaida. Que este se encienda cuando no tengamos tensión. Debido a que si llega a haber una falla en el sistema de la lógica puede correr peligro la seguridad de la gente, por lo que por defecto activamos el paracaidas.



Usar Lógica negativa por la misma razón que antes. Si el sistema no funciona la alerta queda encendida por más que nadie la apriete.



Lás listas Ordenadas acá serían fáciles de armar ya que al armar el ascensor sabremos cuantos pisos tenemos o podemos tomar un máximo y armarlo en un simple vector con posiciones fijas. De esta forma ahorramos en codificación para el ordenamiento. Acá habría que analizar si conviene utilizar RAM asignada a Código por ahorrarme un poco de memoria de datos o viceversa. Un hecho que si sería importante es que para encontrar que botón es utilizaría una serie de IF anidado y no un bucle while con un IF adentro que vaya cambiando una variable. De esta forma escribimos más lineas de código pero ahorramos una variable y cuando se ejecute el programa no va a estar realizando saltos en la memoria para ir de un sector al sector anterior, los cambios de las variables que tiene que hacer de por medio, en total es más trabajo en assembly para la máquina.