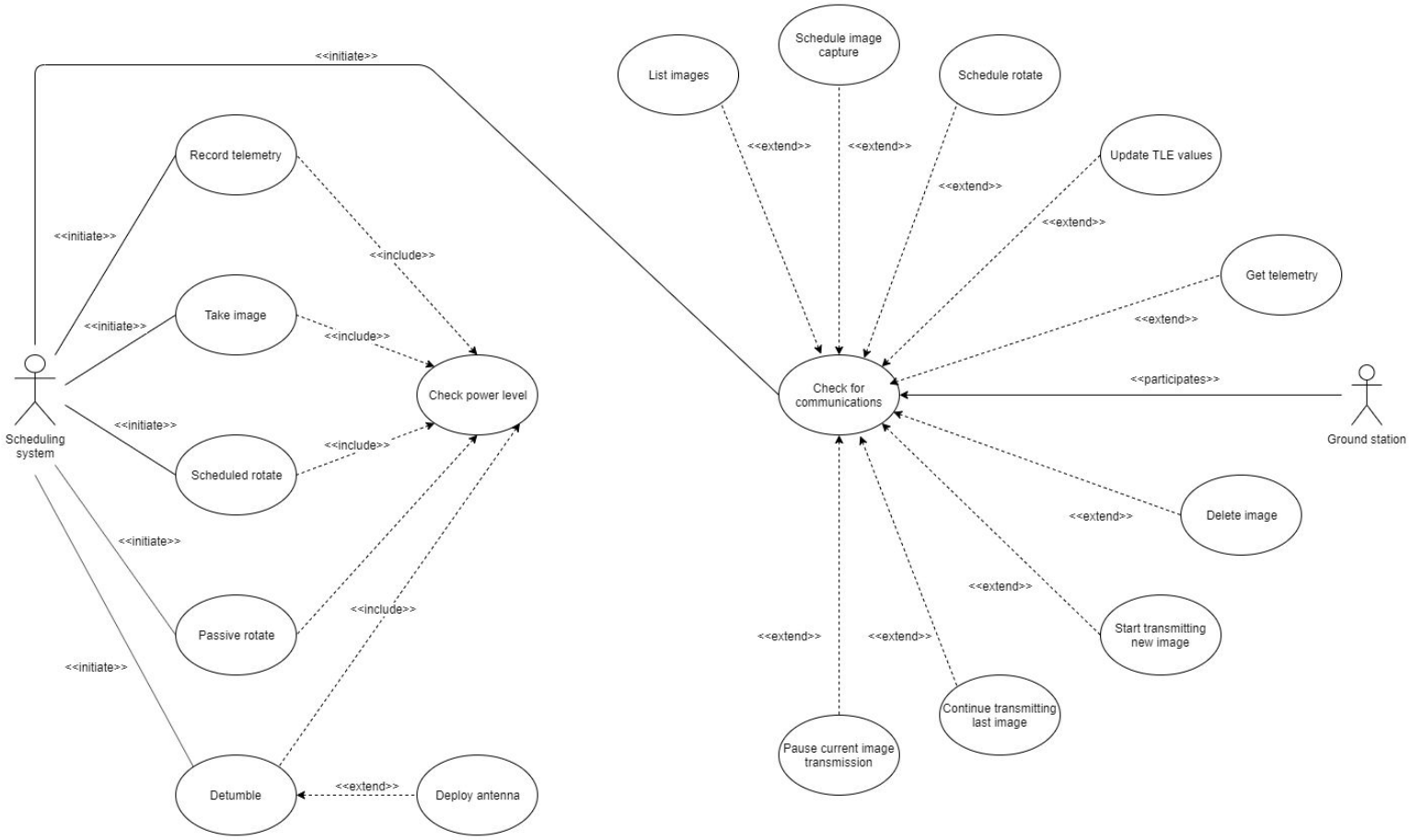

Scheduling system

All automatic/timed/queued checks/tasks by the scheduling system. Entry conditions should probably just be a power level check beforehand.

Use case name	Record telemetry
Participating actors	Scheduling system
Entry conditions	Once the scheduled time to record a specific piece of telemetry is reached, the scheduler will initiate this function if the SD card has enough space and the system power is above a specified value.
Flow of event	1. Query each sensor and subsystem via UART, SPI protocols, etc. and store them in variables. 2. Record the data to a text file in comma-delimited? format. 3. Save the data to the SD
Exit conditions	Data has been recorded to its log file on the SD
Quality requirements	Readable data: data should be formatted in the standard recognized by the system. Reliability: The system should raise an alert if a failure occurs and retry the function. Accuracy: All data to remain accurate.

Unknown:

- What format should the data be recorded in?
- 1 file or multiple files (since different parts record at different rates)?

I took a look at the aggregator files but just from that its not entirely clear.

Use case name	Take image
Participating actors	Scheduling System
Entry conditions	Once the command to take an image is received (in the correct format), the scheduler will initiate this function if the SD card has enough space and the system power is above a specified value.
Flow of event	1. Send command via UART to camera to capture image 2. Read the camera buffer via UART and write it to an image file on the SD card using the timestamp as the image name
Exit conditions	Image has been written to the SD card
Quality requirements	Reliability: Should attempt a second capture or SD card write if it fails the first time Functionality: Correct resolution Usability: Compressed to specified size

Is using the timestamp/date as the file name appropriate?

Use case name	Check power level
Participating actors	Scheduling System
Entry conditions	This check is invoked by another task that includes it, typically as its entry condition.
Flow of event	1. Query the power level of the EPS 2. Return the power level
Exit conditions	N/A
Quality requirements	Accuracy: Recorded amount of power left should be an accurate measure.

Unsure of how the communication between the OBC and EPS will work. Will it occasionally send data on its own, or must we query it first? It uses UART to communicate with the OBC so assuming it just sends it on its own -> What format? This info isn't in either the OBC or EPS data sheets so either need to get the module/email them.

Use case name	Scheduled rotate
Participating actors	Scheduling system
Entry conditions	1. The specified time of a scheduled rotate task has been reached 2. Power levels are sufficient, obtained from the extended Check Power use case
Flow of event	1. Generate fake voltage data (normally based on the argument direction). 2. Pass the fake data via UART to the attitude control board.
Exit conditions	The data has been successfully passed to the attitude control board
Quality requirements	The direction and time should be supplied to the function in an appropriate format

Using freeRTOS: permanent task for reading rotate commands, e.g. from a queue? Or dynamically create/delete tasks when rotate/image commands are received.

Queue: Order commands received in matters, but can easily clear image/rotate command queue.
Dynamic tasks: Order doesn't matter, but managing them is more complex. Significantly more heap space usage. (e.g. create task then vTaskDelay(time to timestamp))

Both methods won't 'survive' a restart but assuming image/rotate commands are executed relatively soon after a command is scheduled.

Use case name	Passive rotate
Participating actors	Scheduling system
Entry conditions	1. The CubeSat is not currently commanded to hold a specific attitude. 2. Power levels are sufficient, obtained from the extended Check Power use case
Flow of event	1. Generate fake voltage data 2. Pass this data to the attitude control board.
Exit conditions	The data has been successfully passed to the attitude control board
Quality requirements	N/A

How it should probably work (but we don't have the sensors or attitude control board yet):

- Read e.g. gyroscope data for current rotation, determine what voltages we need to modify our rotation into the one desired (e.g. around z-axis of CubeSat).
- Pass this data to the attitude control board.

Use case name	Detumble
Participating actors	Scheduling System
Entry conditions	N/A (executed on startup)
Flow of event	1. Generate fake voltage data (normally based on a DEFAULT direction). 2. Pass the fake data via UART to the attitude control board. 3. Once the CubeSat has detumbled, check the status of the antennas a. The Deploy Antenna use case is extended here
Exit conditions	Antennas' deployment statuses are checked and their deployment attempted if need be.
Quality requirements	Tumbling decreased to less than a specified value, less than 1 failure every 100 attempts, priority over take image and rotate.

Will probably detect that it has detumbled via onboard gyroscopes. So will use sensor calls to that, or perhaps the attitude control board will handle this by itself? Will need to ask.

This can probably just be a run-once task with priority second to the interrupt/task for receiving commands. As such it should run immediately.

If we have low power, just let it tumble for a while?

Use case name	Deploy antenna
Participating actors	Scheduling system
Entry conditions	This use case extends the Detumble use case. It is initiated at the end of the detumble use case if the antenna is not deployed.
Flow of event	1. Send current through the burn-wire pins of all antennas. 2. Once all antenna deployment sensors report that the antennas are deployed, stop providing current to the burn-wire pins
Exit conditions	The antennas are deployed
Quality requirements	N/A

We don’t currently know where exactly this will interface.

COMMUNICATIONS

First UC is the ‘check for comms’, which extends the rest for the specific commands, e.g. list images, delete images, transmit telemetry, etc.

Commands that cause transmission back to the ground station will have to format their data in the discussed AX.25 packet format.

Use case name	Check for communications
Participating actors	Scheduling system, transceiver (from GS)
Entry conditions	The transceiver has sent data to the OBC
Flow of event	1. The transceiver sends data via UART to the OBC 2. The data is parsed for commands and parameters 3. The relevant use case is extended here
Exit conditions	The command is successfully executed
Quality requirements	Multiple commands and parameters should be able to be sent in one string

STM32 can generate interrupts on receiving data from UART (tested and it works fine, can use priority levels also), so in that interrupt we can just parse the data and do whatever command/function in there that we need to.

Use case name	List images
Participating actors	Scheduling system, transceiver (from GS)
Entry conditions	This use case extends the Check for communications use case. It is initiated if the received command is to ‘list images’.
Flow of event	1. Query the SD for all the names of available images 2. The list of names is sent to the transceiver via UART
Exit conditions	List of names is successfully sent to the transceiver
Quality requirements	Readability:

Use case name	Schedule Image Capture
Participating actors	Scheduling system, transceiver (from GS)
Entry conditions	This use case extends the Check for communications use case. It is initiated if the received command is to ‘schedule an image capture’.
Flow of event	1. Insert into the schedule to capture an image at the given timestamp
Exit conditions	The image capture has been scheduled
Quality requirements	Command to take image is in the correct format, task has an appropriate interrupt protocol.

Possible ideas discussed above using queues or dynamic tasks.

Use case name	Schedule Rotate
Participating actors	Scheduling system, transceiver (from GS)
Entry conditions	This use case extends the Check for communications use case. It is initiated if the received command is to ‘schedule a rotation’.
Flow of event	1. Insert into the schedule to rotate at the given timestamp
Exit conditions	The rotation has been scheduled
Quality Requirements	Command to make CubeSat rotate is in the correct format

Possible ideas discussed above using queues or dynamic tasks.

Use case name	Update TLE Values
Participating actors	Scheduling systems, transceiver (from GS)
Entry conditions	This use case extends the Check for communications use case. It is initiated if the received command is to ‘update TLE values’.
Flow of event	1. Update TLE values with the data parsed by check communications
Exit conditions	TLE values are updated; if failed, updating is attempted once more.
Quality requirements	Command to update TLE is in the right format, task has an appropriate interrupt protocol.

Use case name	Send Telemetry
Participating actors	Scheduling systems, transceiver (from GS)
Entry conditions	This use case extends the Check for communications use case. It is initiated if the received command is to ‘send telemetry’.
Flow of event	1. The specified telemetry data is read from file 2. Telemetry data is sent to the transceiver via UART
Exit conditions	Telemetry data is sent to the transceiver; if failed, sending is attempted once more.
Quality requirements	The telemetry is sent in an appropriate, minimal data format, task has an appropriate interrupt protocol.

Use case name	Delete Image
Participating actors	Scheduling systems, transceiver (from GS)
Entry conditions	This use case extends the Check for Communications use case. It is initiated if the received command is to delete an image given an image id/name.
Flow of event	1. Query the SD for an image id that equates an image id passed into the function. 2. Select the image. 3. Remove the image details from the SD.
Exit conditions	A success in deleting image message is passed to the transceiver via UART.
Quality requirements	N/A

Use case name	Start Transmitting New Image
Participating actors	Scheduling systems, transceiver (from GS)
Entry conditions	Extended from check communications; will execute if certain command is received
Flow of event	1. Query the SD for an image id that equates an image id passed into the function 2. Select the image 3. Record the transmission details to the SD card (file name, progress(initially 0%)) 4. Start transmission of the new image to the transceiver via UART
Exit conditions	1. Finished transmitting an image 2. Pause transmission of the current image
Quality requirements	Command to start new transmission is in the right format, task has an appropriate interrupt protocol.

Use case name	Continue Transmitting Last Image
Participating actors	Scheduling system, transceiver (from GS)
Entry conditions	1. Extended from check communications; will execute if certain command is received 2. A transmission details file contains info on an image to continue transmitting (file name, progress)
Flow of event	1. Select the image 2. Start transmitting the bytes left to the transceiver via UART
Exit conditions	Image transmission is complete, image transmission is paused, or image transmission failed
Quality requirements	Command to continue transmission is in the right format, task has an appropriate interrupt process.

Use case name	Pause Current Image Transmission
Participating actors	Scheduling system, transceiver (from GS)
Entry conditions	Extended from check communications; will execute if pause command is received; if ground station is out of range during an image transmission
Flow of event	1. Locate how many bytes of the image is left to be transmitted 2. Pause transmission of the image 3. Store the amount of bytes left over to transmit onto the SD
Exit conditions	Progress is stored successfully
Quality requirements	Bytes left over to complete transmission are stored in the correct format

These are what we think may work for the commands, however may be easily changed if needed. Is this an appropriate way to conduct the image transmissions? E.g. GS sends us a command to start transmitting, and it also sends us a command telling us to stop.

Additionally, if the CubeSat is waiting for the GS to make contact with it (rather than the cubesat predicting when to transmit), will we have to worry about filtering incoming data (e.g. from some source other than the GS?)

Additionally additionally, will the data from the GS be encrypted/the CubeSat require some sort of identification/authentication before taking commands? I believe that transmission TO the CubeSat may be encrypted, but not from the CubeSat to GS.