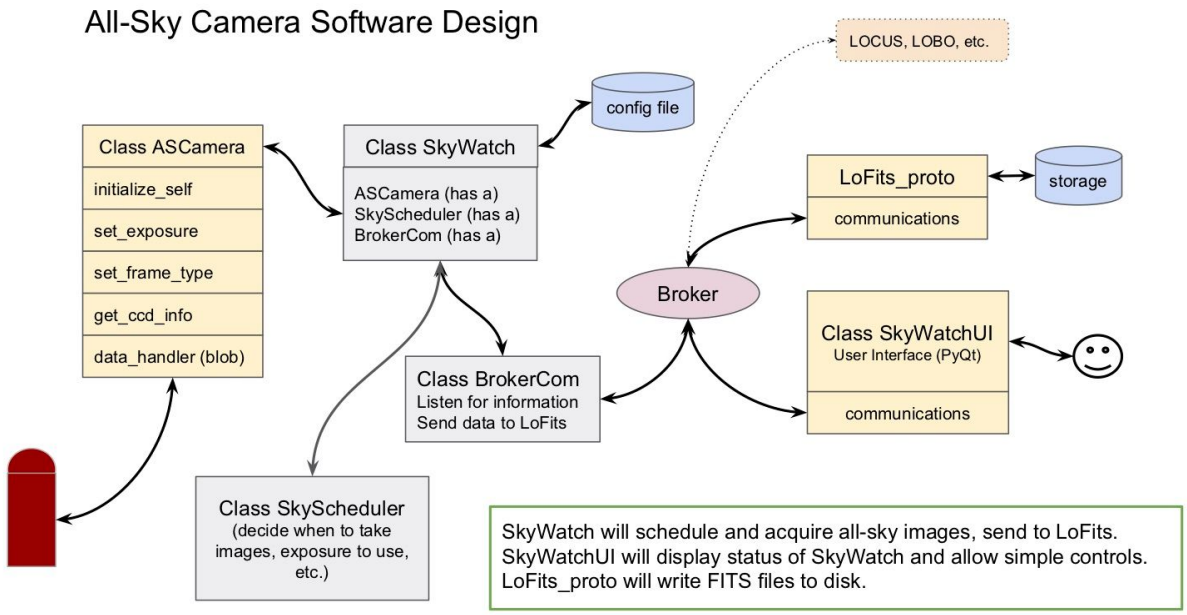
SkyWatch Design Ideas

This document describes Dyer's ideas for the part of the all-sky camera software that acquires images from the camera and writes them as FITS files. I'm calling the program "SkyWatch" for lack of a better name, this is only a placeholder for now. I'll refer to the diagram below.



This diagram is derived from a diagram made by Len Bright that is in the All-Sky GitHub repository with the name "allsky_design_chart.png".

The software will consist of three separate programs/processes that can be running on the same computer or different computers.

- The main SkyWatch program. This is the main program that acquires the images from the camera and sends them to the FITS writing routine.
- The LoFits prototype. This is a program that gathers information from the ActiveMQ broker, listens for FITS file requests, and constructs and writes FITS files to storage.
- The SkyWatch User Interface. This program is used to monitor the status of SkyWatch and provided a method for users/TOs to start, stop, and restart the program when needed.

Main SkyWatch program

The main SkyWatch program will read a configuration file at startup. SkyWatch will have a scheduler object that keeps track of the time and sky brightness and will determine when images should be taken and at what exposure time. It may also schedule times when BIAS frames and DARK frames can be taken, perhaps requiring the camera be covered for these. SkyWatch will send images acquired from the camera to the LoFits prototype to be written to storage.

SkyWatch will regularly send status updates to the the SkyWatch user interface and listen for commands from the same interface.

Some thought should be given to the object oriented structure of this program. One of the objects should be the “camera”, another could be the “scheduler”, and a third could be the communications object used to interact with the ActiveMQ broker.

LoFits Prototype

This program will monitor certain topics on the ActiveMQ broker and keep a store of information needed for the FITS headers up to date.

It will also listen for requests to write a FITS file on an assigned broker topic. When it receives such a request along with a minimal header and data blob, it will construct an up to date FITS header and write it to the directory specified in the minimal header from SkyWatch.

The object structure of this program needs some thought. I will need a main program, perhaps a broker communications object, and perhaps a data aggregator object.

SkyWatch User Interface

SkyWatchUI will be a PyQt program that can be run by the Telescope Operators (and perhaps others) that will allow them to keep track of the status of the SkyWatch program, stop SkyWatch, and restart SkyWatch program.

The object structure of this user interface should be quite simple, perhaps with only a main program and a broker communications object.

Communication Protocols

Communications between various pieces of the system will be very important as they will eventually be generalized to the larger LOCUS system. Topic names will need to be decided upon and some thought given to queues and whether we might need them in certain circumstances. The Broker and Stomp allow various header types and data types as well as acknowledgement or “acks” and no acknowledgement or “nacks” and we’ll need to look at what pieces we need to use and when.