Sky Imager Aggregator Documentation for Programmer

The system consists of 3 python3 scripts. Two executable scripts: SkyImagerV2.py (download images from camera and try to send to remote server) and SendStorageV2.py (send the images that for some reason were not sent on time) and the library LibrforPiV2.py written for the needs of the program. Systemd controls the scripts execution.

The main reason we have two executable scripts is because python has problems with multiprocessing. It doesn't use different cores for different processes instead swaps between the two processes. So to solve this issue we use two scripts that work in different times.

Both scripts are controlled by systemd. SkyImagerV2 is set up to start automatically at power up. In the case of a crash the program would auto-restart.
Systemd executes SendStorageV2 every hour.

LibrforPiV2.py

This python3 script represents the library that contains all of the functions we need. It starts by importing some of the other libraries needed which are: numpy, OpenCV, datetime, http and others.

Function Documentation

def LibrforPiV2.get_SunR_SunS (camera_latitude, camera_longitude, camera_altitude, print_time, date = dt.datetime.now(dt.timezone.utc).date())

Functon calculates sunrise and sunset.

Parameters:

in	*camera_latitude*	camera position - latitude
in	*camera_longitude*	camera position - longitude
in	*camera_altitude*	camera position - altitude
in	*print_time*	unused

in	*date*	day in which sunrise und sunset calculates

Returns:

sunrise and sunset datetime

def LibrforPiV2.hmac_sha256 (message, key)

Calculate keyed - hash for communication authentication.

Parameters:

in	*message*	text string to be hashed
in	*key*	hash key

Returns:

hash

def LibrforPiV2.http (url, data)

Sends POST request.

Parameters:

in	*url*	destination url
in	*data*	sending data

Returns:

return response object

def LibrforPiV2.maskImg (image, mask_path)

Apply mask to the image OpenCV loads the images as multi-dimensional NumPy arrays but in reverse order: The mask is previously created in matlab in the format bmp or png.

Note:

Image and mask must have same dimension

Parameters:

in	*image*	source image in which we apply the mask
in	*mask_path*	path to black and white mask

Returns:

return masked image

def LibrforPiV2.save_to_storage (img, path, name, logger)

Functon saves image to local storage.

Parameters:

in	*img*	image object to save

in	*path*	path to local storage
in	*name*	name of saved image
in	*logger*	logger object

def LibrforPiV2.set_log_to_file (log_path, log_to_console, logger, console_logger)

Function sets logging to file for given day.

Parameters:

in	*log_path*	path to log files storage
in	*log_to_console*	boolen value to remove log to console
in	*logger*	logger object
in	*console_logger*	console logger handler

Returns:

logging.FileHandler

def LibrforPiV2.set_log_to_file_new_day (log_path, logger, hdlr)

Function creates new log file which is unique for every day.

Parameters:

in	*log_path*	path to log files storage
in	*logger*	logger object
in	*hdlr*	old logging.FileHandler

Returns:

new logging.FileHandler

def LibrforPiV2.set_logger (log_level)

Function creates and initializes logging.

Parameters:

in	*log_level*	level of logger

Returns:

logger object and console logger

def LibrforPiV2.upload_json (image, file_time, server)

Functon prepares and sends data to server.

Data send in JSON format

Parameters:

in	*image*	masked image intended for sending
in	*file_time*	image file time
in	*server*	remote server url

Returns:

respose of server

LibrforPiV2.config_obj Class Reference

class consist of configuration variables of application that are read from config.ini

Public Member Functions

- def **__init__** (self, path_config, logger)

Public Attributes

- **cap_url**
- **path_storage**
- **server**
- **log_path**
- **log_to_console**
- **upload_format**
- **camera_latitude**
- **camera_longitude**
- **camera_altitude**
- **debug_mode**
- **filetime_format**
- **image_quality**
- **crop**
- **mask_path**
- **cap_mod**
- **added_time**

SkyImagerV2.py

The SkyImagerV2.py program does the following:

1. Access the camera

This is done by accessing the camera via url. Since the RaspberryPi is installed in a remote location and it probably will not have a display attached to it, camera stream displaying is not available.

2. Taking a picture, apply mask and encode to jpeg format

Picture is after download from camera crop according to values stored in config.ini. Then is applied black and white mask to decrease size in jpg image format. Encoding to jpg format follows.

3. An attempt to send an image to the server

After encoding image to jpeg is image encode to base64 in order to store at JSON structure which is required by remote server. It will increase image size by 33% and it will be good to use another communication format.
If for some reason script fails the code then continues to storing the image and logs error message.

Function Documentation

def SkyImagerV2.add_Image_job (sched, conf, logger, date = dt.datetime.now(dt.timezone.utc).date())

functon adds job to scheduler object for given day Job starts at sunrise and finish at sunset

Parameters:

in	*sched*	apscheduler object
in	*conf*	object with configuration
in	*logger*	logger object
in	*date*	date of jobs, default value is today

def SkyImagerV2.control_job (sched, conf, logger)

auxiliary function that checks whether the main job is running

Parameters:

in	*sched*	apscheduler object
in	*conf*	object with configuration
in	*logger*	logger object

def SkyImagerV2.get_job_parametr (job)

Creates text string with information about given job for save to log file.

Parameters:

in	*job*	job object

Returns:

text string with information about job

def SkyImagerV2.main ()

entry point of script that create main and auxiliary job

def SkyImagerV2.processImage (sched, conf, logger)

owns core of the script that gets image from camera and sends to remote server

Parameters:

in	*sched*	apscheduler object
in	*conf*	object with configuration
in	*logger*	logger object

SendStorageV2.py

SendStorageV2.py is a script created to send the images that for some reason were not sent on time. It takes images from local storage and tries to send to remote server