Etc

util.py

```
from datetime import datetime

class Util:
	def log(source, msg):
		print(datetime.now().strftime("%Y%m%d_%H%M%S") + " | " + source + " | " + str(msg))
```

lock_manager.py

```
import os
import datetime

class Lock_Manager:
	def __init__(self, name):
		self.locks = os.path.dirname(os.path.realpath(__file__)) + "/locks"
		self.name = name + ".lock"
		self.path = os.path.join(self.locks, self.name)

		if not os.path.exists(self.locks):
			os.makedirs(self.locks)

	def set(self):
		if not os.path.exists(self.path):
			print("Setting {} lock".format(self.name))
			timestamp = datetime.datetime.now().strftime("%Y%m%d_%H%M%S")

			with open(self.path, 'w+') as f:
				f.write(timestamp)

	def remove(self):
		if os.path.isfile(self.path):
			os.remove(self.path)

	def get_lock_time(self):
		for filename in os.listdir(self.locks):
			with open(self.locks + "/" + filename) as f:
				return f.read()

		return None
```

conf.py

```
from pathlib import Path

p = Path("logs")
if not p.exists():
   p.mkdir()

dictConfig = {
   'version': 1,
   'disable_existing_loggers': True,
   'formatters': {
      'standard': {
         'format': '%(asctime)s [%(levelname)s] %(name)s:: %(message)s',
      },
```

```
    },
    'handlers': {
        'default': {
            'level': 'DEBUG',
            'formatter': 'standard',
            'class': 'logging.StreamHandler',
            'stream': 'ext://sys.stdout',
        },
        'file': {
            'class': 'logging.handlers.RotatingFileHandler',
            'level': 'DEBUG',
            'formatter': 'standard',
            'filename': 'logs/logfile.log',
            'mode': 'a',
            'maxBytes': 5_242_880,
            'backupCount': 3,
            'encoding': 'utf-8',
        },
    },
    'loggers': {
        '__main__': {
            'handlers': ['default','file'],
            'level': 'DEBUG',
            'propagate': False,
        },
        'camera': {
            'handlers': ['default', 'file'],
            'level': 'DEBUG',
            'propagate': False,
        },
    }
}
```

power.py

```
from vcgencmd import Vcgencmd

vcgm = Vcgencmd()

class PowerStatus(object):
    def __init__(self):
        print("> vcgm.version()=" + vcgm.version())

    def getPowerStatus(self):
        output=vcgm.get_throttled()
        print("raw_data=" + output['raw_data'])
        print("binary=" + output['binary'])
        return "raw_data=" + output['raw_data'] + ", binary=" + output['binary']
```

Revision #1
★Created 1 June 2023 14:41:35 by Hyeon Su Ryu
Updated 1 June 2023 14:43:27 by Hyeon Su Ryu