Improved documentation

- Short documentation (usage description) for *gitio* has been created.
- *Gitio* is our tool for managing Git submodules.

Prerequisites

- Git Bash on Windows has to be installed with 'support for symlink files' option enabled
- Script needs to work on non-interactive repositories (i.e. repositories cloned via ssh key)

Usage

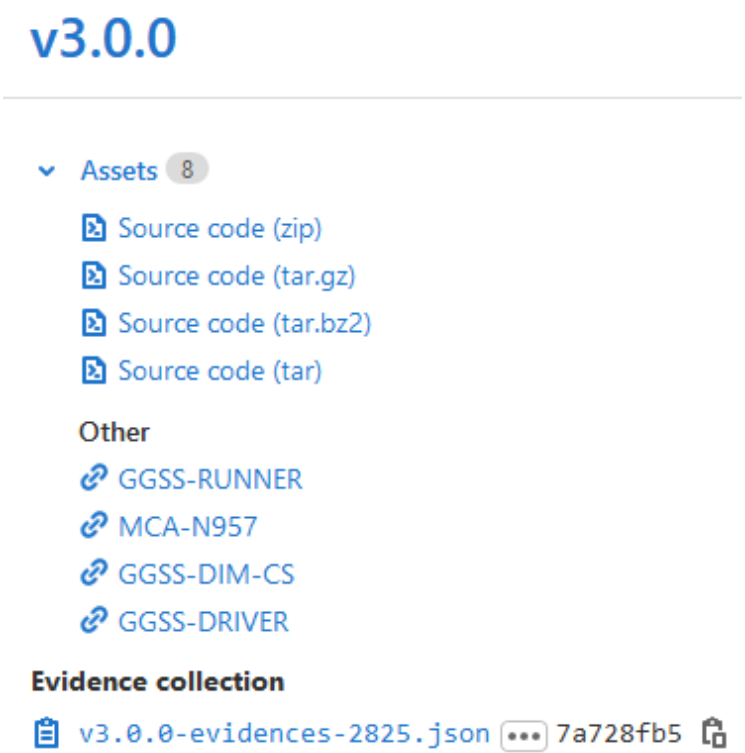
```
usage: gitio.py [-h] -p PATH [-b BIN]

Application for multi-level recursive repository handling

optional arguments:
  -h, --help            show this help message and exit
  -p PATH, --path PATH  Path to root of the repository tree (master
                        repository)
  -b BIN, --bin BIN     Path to git binary (needed if git binary is not
                        automatically detected)
```

Notes

1. Before using gitio make sure to initialize submodules - `git submodule update --init --recursive --remote`
2. BIN parameter is only needed if gitio does not automatically detect GIT binary.

Custom artifacts for releases

- Automated versioning has been improved and finished.
- Compiled applications (and driver) are now part of generated releases.
- They can now be downloaded directly from GitLab.



HV management commands refactoring

- More user-friendly commands for HV management using DIM.
- GET (MON) and SET command changed (no need to send raw HV commands).
- Example syntax: `hv 0:* mon vmon`.
- `*` - all channels (or modules).
- Still work in progress.

Current plans

- Single RPM with all GGSS applications and utilities (for fast and easy deploy).
- Hardware testing scripts refactoring and upgrade (yaml based scenarios).
- Additional documentatiom improvements (GitLab CI tasks debugging).
- Further code refactoring.
- GGSS improvements, for example peak finding / range fitting.