Overview

The purpose of this three part series is to give an overview of how SATnogs communicates with rotators and how to build a CANopen controlled rotator. The European Space Agency has released a standard for the use of CANopen in satellites. The purpose of using CANopen in our rotator is to gain familiarity with the protocol.

Part 1 – How SATnogs talks to rotators
Part 2 – How CANopen works and how to use CANopen with SATnogs
Part 3 – How to interface a motor controller to the CANopen bus

Part 1: How SATnogs talks to rotators

SATnogs talks to rotators thru the hamlib library (https://github.com/Hamlib/Hamlib/wiki).

Hamlib allows control of radios and rotators thru a unified API. We are mainly interested in rotctld which is used for rotator control. Rigctld is used to command gnuradio settings in SATnogs.

Rotctld is a daemon that handles the communication between rotators and applications that control rotators. Rotctld has support for different rotators. Each rotator type has an id associated with it. The list can be found here https://github.com/Hamlib/Hamlib/wiki/Supported-Rotators

SATnogs-client communicates with hamlib using rotctl protocol. The commands used by rotctl are documented here http://hamlib.sourceforge.net/manuals/hamlib.html#rotctld-commands

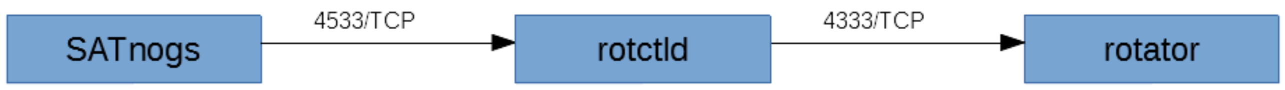
SATnogs-client first reads the current position of the rotator and after that sends a new az/el that the rotator should move to. https://gitlab.com/librespacefoundation/satnogs/satnogs-client/blob/master/satnogsclient/observer/worker.py#L146

Here is an example of the communication to between SATnogs and a hamlib compatible rotator

```
** ** ** ** **
p\n                      // SATnogs requests the current position of the rotator, the
                         // command is ended with a newline \n

33.000000\n33.000000\n   // Rotator responds with the azimuth and elevation of the rotator. Each
                         // value is separated by \n and response is ended with a newline \n

P 13.9709333812 1.67219498041   // SATnogs sends a command to move the rotator to a new
                                // position.

RPRT 0                   // Rotator responds with return code. If everything was ok, the code is 0
** ** ** ** **
```

For commands that are not implemented the rotator should respond with RPRT -4.

The rotator type is determined when starting rotctld. SATnogs rotator uses the EASYCOMM III protocol but for us NET rotctl might be better. It uses TCP sockets instead of a serial port.



For manual testing rotctld can be started with the NET rotctl rotator on the same computer on port 4333. The rotator server needs to be running before starting rotctld.

```
rotctld -m 2 -r 127.0.0.1:4333
```

By default rotctld uses port 4533 for listening to control applications. If you are already running another instance of rotctld you probably need to change the listening port with -t. Also adding -vvv will give you more information to debug the possible errors.

```
rotctld -m 2 -r 127.0.0.1:4333 -t 4000 -vvv
```

nc can be used to send commands to rotctld when manually testing

```
nc localhost 4000
```

\dump_state is a special command that rotctld sends when connecting to the rotator using the NET rotctl backend. It queries the rotator for capabilities but all of the fields are for rigctl so it isn’t that useful with only rotators.