



# TERMITE OS

THE TELESCOPE OPERATING SYSTEM

## **termiteOS Documentation**

*Release alpha*

**Nacho Mas**

July 30, 2018



## CONTENTS

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Architecture</b>	<b>5</b>
<b>3</b>	<b>License</b>	<b>7</b>
<b>4</b>	<b>Technical information</b>	<b>9</b>
	<b>Index</b>	<b>25</b>



A telescope Operating System base on zmq and Protocol Buffers. Documentation <https://nachoplus.github.io/termiteOS/index.html>



## INTRODUCTION

Termite OS is a modular, adaptable, easily extendable telescope operating system developed primarily, but not exclusively, in python.

It wants to answer to several limitations that most of the commercial mount controllers have. It wants to be a platform where you can incorporate all kinds of functionalities that the professional or amateur astronomer may need in a simple way.

Possible use case:

- To motorize a DIY mount or retrofit a old one.
- Make your telescope be able to track the ISS or any other satellite.
- Implement new protocol commands.
- New native protocols, i.e. indilib
- Control over the objects catalogs built into your telescope.
- Integrate pointing model in your mount.
- Connect a GPS to your mount and use his data for location and time.
- Define the horizon of your observatory.
- WiFi or Bluetooth access to your mount
- Develop new motion strategies.

With termiteOS you will address all this things and more.

Termite OS is a **work in progress** but much of the functionality is already available:

- Stepper controller using Raspberry PI and the integrated DRV8825 widely used in the 3D printer world
- LX200 command set
- Slew and celestial track
- Satellite tracking

Ongoing functionality:

- Arduino base hardware
- BLDC motors
- Servo motors
- Web interface
- Constellation pointing





## **ARCHITECTURE**

Each termiteOS functionality is implemented as a separate program called a 'node'. The nodes communicate each other using the zmq protocol. The organization between the nodes is hierarchical thus a node can have several children but has only one parent or none in the case of the 'root node'.

ZMQ (<http://zeromq.org/>) is used for transport and on <https://developers.google.com/protocol-buffers/> for message definitions and serialization.

Each node has its own ZMQ port and a set of commands and responds to through that port. Each node opens connections with its parent node and with all its children so that messages can be exchanged.

These nodes can run on the same or different CPUs taking advantage of all the features of the ZMQ protocol.



---

**CHAPTER  
THREE**

---

**LICENSE**

GPL3 Copyright (c) July 2018 Nacho Mas

Logo made with <https://www.designevo.com/en/> DesignEvo



## TECHNICAL INFORMATION

### Launcher

- **Launcher** program launch a set of node following the instruction in yaml file

This program allow to launch at once all nodes needed for a specific hardware/funtionality.

Example of a yaml configuration file:

```
simple:
  type: telescope
  host: localhost
  port: 5000
  nodes:
    - LX200:
      type: tcpproxy
      host: localhost
      port: 5001
      params: {'tcpport':6001,'End':'#'}
    - tracker:
      type: TLEtracker
      host: localhost
      port: 5002
      nodes:
        - trackertcp:
          type: tcpproxy
          host: localhost
          port: 5003
          params: {'tcpport':6003}
```

This example is equivalent to run on the command shell all following commands:

```
miteTelescope --port 5000 --name simple
mitetcpproxy  --port 5001 --name LX200 --parent_host localhost --parent_port 5000 --params {'tcpport':6001,'End':'#'}
miteTLEtracker --port 5002 --name tracker --parent_host localhost --parent_port 5000
mitetcpproxy  --port 5003 --name trackertcp --parent_host localhost --parent_port 5002 --params {'tcpport':6003}
```

You can find other examples in ‘termoteOS/machines/’

### Command line:

#### miteLaunch

Launch all nodes defined in YAMLFILE

```
miteLaunch [OPTIONS] YAMLFILE
```

## Arguments

### YAMLFIELD

Required argument

## API

Launch tools to run a rig. Run several daemons at once

`termiteOS.launch.launchmachine (yamlfile)`

Launch an arrangement of daemons defined in a yaml file

`termiteOS.launch.launchnode (nodedict, parent_host='', parent_port=False)`

Launch a node defined in dictionary. Called recursively

`termiteOS.launch.run_in_separate_process (func, *args, **kws)`

Run function in a separate process. To background executables

## Nodes

Nodes are separate programs. The nodes communicate each other using the zmq protocol. The organization between the nodes is hierarchical thus a node can have several children but has only one parent or none in the case of the 'root node'.

Nodes are based on <http://zeromq.org/> for transport and on <https://developers.google.com/protocol-buffers/> for message definitions and serialization.

Each node has its own ZMQ port and a set of commands and responds to through that port. Each node opens connections with its parent node and with all its children so that messages can be exchanged.

These nodes can run on the same or different CPUs taking advantage of all the features of the ZMQ protocol.

All nodes are derive from nodeSkull base class which implements all the communication logic and basic commands.

## TLEtracker

- **TLEtracker** node calculate satellite TLE speed and RA/DEC and send to the mount

With this node the mount is able to follow any object with TLE.

### Command line:

#### miteTLEtracker

Launch a TLEtracker node

```
miteTLEtracker [OPTIONS]
```

### Options

**--name** <name>  
module name

**--port** <port>  
Port listen

**--parent\_host** <parent\_host>  
Parent host to connect to. If False the node become a ROOTHUB

**--parent\_port** <parent\_port>  
Parent port to connect to

## TLTracker commands

### follow

Follow satellite

### help

Print help text. Do nothing. Normaly overloaded by a child class

### ls

list the node commands

### nodes

list the children nodes:

**param arg** (None)

**returns** a python list all children node names

### ping

None

### tree

Print all self and children nodes availabled commands

**param arg** (None)

**returns** a string contain all commands

## hub

- **hub** node has not own commands. Only used to connect other nodes

## Command line:

### miteHub

Launch a hub node

```
miteHub [OPTIONS]
```

## Options

**--name** <name>  
module name

**--port** <port>  
Port listen

**--parent\_host** <parent\_host>  
Parent host to connect to. If False the node become a ROOTHUB

**--parent\_port** <parent\_port>  
Parent port to connect to

## hub commands

### help

Print help text. Do nothing. Normaly overloaded by a child class

### ls

list the node commands

### nodes

list the children nodes:

**param arg** (None)

**returns** a python list all children node names

### ping

None

### tree

Print all self and children nodes availabled commands

**param arg** (None)

**returns** a string contain all commands

## joystick

- **joystick** node to manual move the mount with a joystick

### Command line:

#### miteJoy

Launch a joystick node

```
miteJoy [OPTIONS]
```

### Options

**--name** <name>  
module name

**--port** <port>  
Port listen

**--parent\_host** <parent\_host>  
Parent host to connect to. If False the node become a ROOTHUB

**--parent\_port** <parent\_port>  
Parent port to connect to

## joystick commands

### help

Print help text. Do nothing. Normaly overloaded by a child class

### ls



list the node commands

### nodes

list the children nodes:

**param arg** (None)

**returns** a python list all children node names

### ping

None

### tree

Print all self and children nodes available commands

**param arg** (None)

**returns** a string contain all commands

## tcpproxy

- **tcpproxy** node acts as a proxy connector between other node and an especific TCP port

All commands recived throught the TCP port are relay to the myCmdPort port of the node conected to.

Using this node allow us to connect to a specific node (the parent node of tcpproxy node) with a regular *telnet host port*

### Command line:

#### mitetcpproxy

Launch a tcpproxy node

```
mitetcpproxy [OPTIONS]
```

### Options

**--name** <name>  
module name

**--port** <port>  
Port listen

**--parent\_host** <parent\_host>  
Parent host to connect to. If False the node become a ROOTHUB

**--parent\_port** <parent\_port>  
Parent port to connect to

**--params** <params>  
Dictionary with extra parameters default={"tcpport":6001,"End": cant be empty}

### tcpproxy commands

#### help

Print help text. Do nothing. Normaly overloaded by a child class

#### ls

list the node commands

#### nodes

list the children nodes:

**param arg** (None)

**returns** a python list all children node names

#### ping

None

#### tree

Print all self and children nodes available commands

**param arg** (None)

**returns** a string contain all commands

## Telescope

- **telescope** node implement a telescope mount basic commands (goto,slew,track..)

Two axis equatorial mount telescope.

### Command line:

#### miteTelescope

Launch a telescope node

```
miteTelescope [OPTIONS]
```

### Options

**--name** <name>  
module name

**--port** <port>  
Port listen

**--parent\_host** <parent\_host>  
Parent host to connect to. If False the node become a ROOTHUB

**--parent\_port** <parent\_port>  
Parent port to connect to

### telescope commands

#### ack

None

#### align2target

None

#### firmware\_date

None

#### firmware\_ver

None

**getLocalDate**

None

**getLocalTime**

None

**getSideralTime**

None

**getTargetDEC**

None

**getTargetRA**

None

**getTelescopeDEC**

None

**getTelescopeRA**

None

**help**

Print help text. Do nothing. Normaly overloaded by a child class

**info**

None

**ls**

list the node commands

**nodes**

list the children nodes:

**param arg** (None)

**returns** a python list all children node names

**ping**

None

**pulseE**

None

**pulseN**

None

**pulseS**

None

**pulseW**

None

**setMaxSlewRate**

None

**setTargetDEC**

None

### **setTargetRA**

None

### **slew**

None

### **slewRate**

None

### **stopSlew**

None

### **tree**

Print all self and children nodes available commands

**param arg** (None)

**returns** a string contain all commands

## **Node commands**

### **TLEtracker commands**

#### **follow**

Follow satellite

#### **help**

Print help text. Do nothing. Normaly overloaded by a child class

#### **ls**

list the node commands

#### **nodes**

list the children nodes:

**param arg** (None)

**returns** a python list all children node names

#### **ping**

None

#### **tree**

Print all self and children nodes available commands

**param arg** (None)

**returns** a string contain all commands

### **costellation commands**

#### **follow**

None

#### **help**

Print help text. Do nothing. Normaly overloaded by a child class

#### **ls**

list the node commands

#### **nodes**

list the children nodes:

**param arg** (None)

**returns** a python list all children node names

#### **ping**

None

#### **tree**

Print all self and children nodes availabled commands

**param arg** (None)

**returns** a string contain all commands

### **hub commands**

#### **help**

Print help text. Do nothing. Normaly overloaded by a child class

#### **ls**

list the node commands

#### **nodes**

list the children nodes:

**param arg** (None)

**returns** a python list all children node names

#### **ping**

None

#### **tree**

Print all self and children nodes availabled commands

**param arg** (None)

**returns** a string contain all commands

### **joystick commands**

#### **help**

Print help text. Do nothing. Normaly overloaded by a child class

#### **ls**

list the node commands

#### **nodes**

list the children nodes:

**param arg** (None)

**returns** a python list all children node names

#### **ping**

None

## **tree**

Print all self and children nodes available commands

**param arg** (None)

**returns** a string contain all commands

## **tcpproxy commands**

### **help**

Print help text. Do nothing. Normaly overloaded by a child class

### **ls**

list the node commands

### **nodes**

list the children nodes:

**param arg** (None)

**returns** a python list all children node names

### **ping**

None

### **tree**

Print all self and children nodes available commands

**param arg** (None)

**returns** a string contain all commands

## **telescope commands**

### **ack**

None

### **align2target**

None

### **firmware\_date**

None

### **firmware\_ver**

None

### **getLocalDate**

None

### **getLocalTime**

None

### **getSideralTime**

None

### **getTargetDEC**

None

### **getTargetRA**

None

**getTelescopeDEC**

None

**getTelescopeRA**

None

**help**

Print help text. Do nothing. Normaly overloaded by a child class

**info**

None

**ls**

list the node commands

**nodes**

list the children nodes:

**param arg** (None)

**returns** a python list all children node names

**ping**

None

**pulseE**

None

**pulseN**

None

**pulseS**

None

**pulseW**

None

**setMaxSlewRate**

None

**setTargetDEC**

None

**setTargetRA**

None

**slew**

None

**slewRate**

None

**stopSlew**

None

**tree**

Print all self and children nodes availabled commands

**param arg** (None)

**returns** a string contain all commands

## Drivers

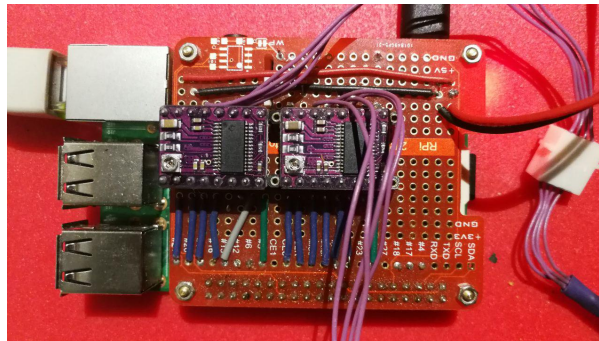
Drivers interact with the real hardware and normaly are used in the **node** code.

### Motors

#### Raspberry Drivers

##### rpiDRV8825Hat

This is a DIY Raspberry Pi Hat base on the popular stepper control chip DRV8825



**API** DIY DRV8825 driver Hat interface.

This board has two DRV8825 able to driver 2 motors See hardware on termiteOS/driver/rpi/hardware

#### INTERFACE TO OTHER MODULES:

- motorBeta
- pinout
- microsteps
- clutch()
- reset()
- sleep()
- set\_microsteps(microsteps)
- sync(motorBeta)

```
class termiteOS.drivers.rpi.rpiDRV8825Hat.rpiDRV8825Hat (raspberrypi, driverID,  
                                                    **kwds)
```

This class define the PIN mapping and basic methods for the rpiDVR8825 Hat

---

**Note:** Possible values of driverID can be 0 or 1.

---

**clutch** (*ON\_OFF*)

Engage or Disengage(free spinnig) the motors

**fault** (*gpio, level, tick*)

Callback function to check internal driver faults



**reset** (*ON\_OFF*)  
Reset the driver circuit

**set\_dir** (*dir*)  
Set the direction of motion

**set\_microsteps** (*microsteps*)  
Set microstepping mode of the driver

**sleep** (*ON\_OFF*)  
Sleep or wake up the driver circuit

**stepcounter** (*gpio, level, tick*)  
Callback function to update internal position counter

**sync** (*position*)  
Set the actual internal position == position

**test** ()  
Test the Hat sending steps

### **rpiSpeedPWM**

A PWM driver

**API** Raspberry PWM motor driver.

INTERFACE:

- **inherits several methods from rpiDRV8825Hut base class**
  - betaMotor
  - pinout
  - microsteps
  - clutch()
  - reset()
  - sleep()
  - set\_microsteps(microsteps)
  - sync(position)
- **Own methods:**
  - setSpeed (radians/seconds)
  - setRPM(RPM)
  - SetPoint(setpoint)
  - goto() -> absolute SetPoint
  - move() -> relative SetPoint
  - stop()
  - isStopped
  - gotoEnd
  - pos

```
class termiteOS.drivers.rpi.rpiSpeedPWM.rpiSpeedPWM (driverID, microsteps, FullTurn-Steps, gear=1, name='Axis', raspberry='localhost')
```

This class do the PWM control calling the underlying pigpiod daemon.

---

**Note:** Up to dates only PID control is implemented.

---

**SetPoint** (*setpoint*)

Establish the \_SetPoint value

**clutch** (*ON\_OFF*)

Engage or Disengage(free spinnig) the motors

**fault** (*gpio, level, tick*)

Callback function to check internal driver faults

**goto** (*setpoint, blocking=False*)

Absolute movement

**gotoEnd**

True if the axis finally arrive to destination(\_SetPoint), False otherwise

**isStopped**

True if is stopped, False otherwise

**move** (*relsetpoint*)

Relative movement

**pos**

Actual position (Corrected motorBeta)

**rampUp** (*v, deltaT, out\_min=-750, out\_max=750*)

Limit motor speed changes to avoid axis stalling

**reset** (*ON\_OFF*)

Reset the driver circuit

**run** (*\*args, \*\*kwargs*)

**setRPM** (*rpm*)

Set and start PWM to obtain rpm

**setSpeed** (*v*)

Set and start PWM to obtain radians/seconds

**set\_dir** (*dir*)

Set the direction of motion

**set\_microsteps** (*microsteps*)

Set microstepping mode of the driver

**sleep** (*ON\_OFF*)

Sleep or wake up the driver circuit

**stepcounter** (*gpio, level, tick*)

Callback function to update internal position counter

**stop** ()

Not implemented

**stopPWM** ()

Stop PWM generation without any check

**sync** (*newposition*)

Establish newposition as current possition (motorBeta)

**test** ()

Test the Hat sending steps

**trackSpeed** (*trackSpeed*)

Set axis track speed. Track speed\*timestep is add to the \_SetPoint value

`termiteOS.drivers.rpi.rpiSpeedPWM.threaded` (*fn*)

Multithread wrapper. Used as a function decorator

## Miscellaneous

TBD

## Indices and tables

- [genindex](#)
- [modindex](#)
- [search](#)



## Symbols

-name <name>  
     miteHub command line option, 11  
     miteJoy command line option, 12  
     mitetcp proxy command line option, 13  
     miteTelescope command line option, 14  
     miteTLE tracker command line option, 10  
 -params <params>  
     mitetcp proxy command line option, 13  
 -parent\_host <parent\_host>  
     miteHub command line option, 11  
     miteJoy command line option, 12  
     mitetcp proxy command line option, 13  
     miteTelescope command line option, 14  
     miteTLE tracker command line option, 10  
 -parent\_port <parent\_port>  
     miteHub command line option, 11  
     miteJoy command line option, 12  
     mitetcp proxy command line option, 13  
     miteTelescope command line option, 14  
     miteTLE tracker command line option, 10  
 -port <port>  
     miteHub command line option, 11  
     miteJoy command line option, 12  
     mitetcp proxy command line option, 13  
     miteTelescope command line option, 14  
     miteTLE tracker command line option, 10

## C

clutch() (termiteOS.drivers.rpi.rpiDRV8825Hat.rpiDRV8825Hat  
     method), 20  
 clutch() (termiteOS.drivers.rpi.rpiSpeedPWM.rpiSpeedPWM  
     method), 22

## F

fault() (termiteOS.drivers.rpi.rpiDRV8825Hat.rpiDRV8825Hat  
     method), 20  
 fault() (termiteOS.drivers.rpi.rpiSpeedPWM.rpiSpeedPWM  
     method), 22

## G

goto() (termiteOS.drivers.rpi.rpiSpeedPWM.rpiSpeedPWM  
     method), 22  
 gotoEnd (termiteOS.drivers.rpi.rpiSpeedPWM.rpiSpeedPWM  
     attribute), 22

## I

isStopped (termiteOS.drivers.rpi.rpiSpeedPWM.rpiSpeedPWM  
     attribute), 22

## L

launchmachine() (in module termiteOS.launch), 10  
 launchnode() (in module termiteOS.launch), 10

## M

miteHub command line option  
     -name <name>, 11  
     -parent\_host <parent\_host>, 11  
     -parent\_port <parent\_port>, 11  
     -port <port>, 11  
 miteJoy command line option  
     -name <name>, 12  
     -parent\_host <parent\_host>, 12  
     -parent\_port <parent\_port>, 12  
     -port <port>, 12  
 miteLaunch command line option  
     YAMLFILe, 10  
 mitetcp proxy command line option  
     -name <name>, 13  
     -params <params>, 13  
     -parent\_host <parent\_host>, 13  
     -parent\_port <parent\_port>, 13  
     -port <port>, 13  
 miteTelescope command line option  
     -name <name>, 14  
     -parent\_host <parent\_host>, 14  
     -parent\_port <parent\_port>, 14  
     -port <port>, 14  
 miteTLE tracker command line option  
     -name <name>, 10  
     -parent\_host <parent\_host>, 10  
     -parent\_port <parent\_port>, 10  
     -port <port>, 10  
 move() (termiteOS.drivers.rpi.rpiSpeedPWM.rpiSpeedPWM  
     method), 22

## P

pos (termiteOS.drivers.rpi.rpiSpeedPWM.rpiSpeedPWM  
     attribute), 22

## R

rampUp() (termiteOS.drivers.rpi.rpiSpeedPWM.rpiSpeedPWM  
     method), 22

`reset()` (`termiteOS.drivers.rpi.rpiDRV825Hat.rpiDRV825Hat` method), 20

`reset()` (`termiteOS.drivers.rpi.rpiSpeedPWM.rpiSpeedPWM` method), 22

`rpiDRV825Hat` (class in `termiteOS.drivers.rpi.rpiDRV825Hat`), 20

`rpiSpeedPWM` (class in `termiteOS.drivers.rpi.rpiSpeedPWM`), 21

`run()` (`termiteOS.drivers.rpi.rpiSpeedPWM.rpiSpeedPWM` method), 22

`run_in_separate_process()` (in module `termiteOS.launch`), 10

`YAMLFILE` `miteLaunch` command line option, 10

## S

`set_dir()` (`termiteOS.drivers.rpi.rpiDRV825Hat.rpiDRV825Hat` method), 21

`set_dir()` (`termiteOS.drivers.rpi.rpiSpeedPWM.rpiSpeedPWM` method), 22

`set_microsteps()` (`termiteOS.drivers.rpi.rpiDRV825Hat.rpiDRV825Hat` method), 21

`set_microsteps()` (`termiteOS.drivers.rpi.rpiSpeedPWM.rpiSpeedPWM` method), 22

`SetPoint()` (`termiteOS.drivers.rpi.rpiSpeedPWM.rpiSpeedPWM` method), 22

`setRPM()` (`termiteOS.drivers.rpi.rpiSpeedPWM.rpiSpeedPWM` method), 22

`setSpeed()` (`termiteOS.drivers.rpi.rpiSpeedPWM.rpiSpeedPWM` method), 22

`sleep()` (`termiteOS.drivers.rpi.rpiDRV825Hat.rpiDRV825Hat` method), 21

`sleep()` (`termiteOS.drivers.rpi.rpiSpeedPWM.rpiSpeedPWM` method), 22

`stepcounter()` (`termiteOS.drivers.rpi.rpiDRV825Hat.rpiDRV825Hat` method), 21

`stepcounter()` (`termiteOS.drivers.rpi.rpiSpeedPWM.rpiSpeedPWM` method), 22

`stop()` (`termiteOS.drivers.rpi.rpiSpeedPWM.rpiSpeedPWM` method), 22

`stopPWM()` (`termiteOS.drivers.rpi.rpiSpeedPWM.rpiSpeedPWM` method), 22

`sync()` (`termiteOS.drivers.rpi.rpiDRV825Hat.rpiDRV825Hat` method), 21

`sync()` (`termiteOS.drivers.rpi.rpiSpeedPWM.rpiSpeedPWM` method), 22

## T

`termiteOS.drivers.rpi.rpiDRV825Hat` (module), 20

`termiteOS.drivers.rpi.rpiSpeedPWM` (module), 21

`termiteOS.launch` (module), 10

`test()` (`termiteOS.drivers.rpi.rpiDRV825Hat.rpiDRV825Hat` method), 21

`test()` (`termiteOS.drivers.rpi.rpiSpeedPWM.rpiSpeedPWM` method), 22

`threaded()` (in module `termiteOS.drivers.rpi.rpiSpeedPWM`), 23