



TERMITE OS

THE TELESCOPE OPERATING SYSTEM

termiteOS Documentation

Release alpha

Nacho Mas

July 28, 2018

CONTENTS

1	Introduction	3
2	Architecture	5
3	License	7
4	Technical information	9
	Index	23

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

API

TLEtracker

class termiteOS.nodes.TLEtracker.**TLEtracker** (*name, port, parent_host, parent_port*)
Command to the parent node to track satellites base on his TLE

HasChildren ()

Return **True** if the node has childrens. **False** otherwise

addCMDs (*CMDs*)
add commands explicetely

circle (*re, dec, r, v*)
Used as test

cmd (*cmd*)
Execute the cmd command

Parameters **cmd** – string contain the command with his parameters

Returns A message containing the answer

cmd_follow (*arg*)
Follow satellite

cmd_help (*arg*)
Print help text. Do nothing. Normaly overloaded by a child class

cmd_ls (*arg*)
list the commands

cmd_nodes (*arg*)
list the children nodes:

Returns a python list with all the children names

cmd_ping (*arg*)

cmd_tree (*arg*)
Print all node and children availabled commands

cmddummy (*arg*)
Default cmd to execute when not knowed cmd match. Do nothing

deregister (*arg*)
Close the zmq socket and deleted node from the children list

end (*arg= ''*)
End all

exenodeCmd (*arg*)
Execute the command in the children

gearInit ()
Get the gear info

heartbeat (*arg*)
heartbeat Parent part

nodeheartbeat (**args, **kwargs*)

observerInit ()
Recover Observer data (lat, lon,...) from parent node

```
register (arg='')  
    Call to parent registrar  
registrar (arg)  
    Registrar parent part  
run ()  
    Main loop. Obtain RA/DEC actual values and do the trackSatellite() call  
satPosition (sat)  
    Calculate satellite RA/DEC from TLE  
satSpeed (sat)  
    Calculate satellite speed from TLE  
scanCMDs ()  
    scanCMD add all methods starting with 'cmd_' as a commands  
sendSlew (RA, DEC)  
    send slew to parent node primitive  
sendTrackSpeed (vRA, vDEC)  
    send speed to parent node primitive  
signal_handler (signal, frame)  
    Capture Ctrl+C key  
trackSatellite (sat)  
    send track speed to parent node. If to far from target send also a slew  
zmqQueue (*args, **kwargs)  
termiteOS.nodes.TLEtracker.runTLEtracker (name, port, parent_host='', parent_port=False)  
ENTRYPOINT calling this fuction start the node
```

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
```


API

Conection HUB

class termiteOS.nodes.hub.hub (*name, port, parent_host, parent_port*)

HasChildren ()

Return **True** if the node has childrens. **False** otherwise

addCMDs (*CMDs*)

add commands explicetely

cmd (*cmd*)

Execute the cmd command

Parameters **cmd** – string contain the command with his parameters

Returns A message containing the answer

cmd_help (*arg*)

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

cmd_ls (*arg*)

list the commands

cmd_nodes (*arg*)

list the children nodes:

Returns a python list with all the children names

cmd_ping (*arg*)

cmd_tree (*arg*)

Print all node and children availabled commands

cmddummy (*arg*)

Default cmd to execute when not knowed cmd match. Do nothing

deregister (*arg*)

Close the zmq socket and deleted node from the children list

end (*arg= ''*)

End all

exenodeCmd (*arg*)

Execute the command in the children

heartbeat (*arg*)

heartbeat Parent part

nodeheartbeat (**args, **kwargs*)

register (*arg= ''*)

Call to parent registrar

registrar (*arg*)

Registrar parent part

run ()

Dummy. Normaly overloaded by a child class

scanCMDs ()

scanCMD add all methods starting with '**cmd_**' as a commands

signal_handler (*signal, frame*)

Capture Ctrl+C key

zmqQueue (**args, **kwargs*)

```
termiteOS.nodes.hub.runhub (name, port, parent_host='', parent_port=False)
```

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
```

API

```
termiteOS.nodes.joystick.runjoystick (name, port, parent_host, parent_port)
```

```
class termiteOS.nodes.joystick.stick (name, port, parent_host, parent_port)
```

```
HasChildren ()
```

Return **True** if the node has childrens. **False** otherwise

```
addCMDs (CMDs)
```

add commands explicetely

```
cmd (cmd)
```

Execute the cmd command

Parameters **cmd** – string contain the command with his parameters

Returns A message containing the answer

```
cmd_help (arg)
```

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

```
cmd_ls (arg)
```

list the commands

```
cmd_nodes (arg)
```

list the children nodes:

Returns a python list with all the children names

```
cmd_ping (arg)
```

```

cmd_tree (arg)
    Print all node and children available commands

cmddummy (arg)
    Default cmd to execute when not knowed cmd match. Do nothing

deregister (arg)
    Close the zmq socket and deleted node from the children list

end (arg='')
    End all

exenodeCmd (arg)
    Execute the command in the children

heartbeat (arg)
    heartbeat Parent part

nodeheartbeat (*args, **kwargs)

register (arg='')
    Call to parent registrar

registrar (arg)
    Registrar parent part

run ()

scanCMDs ()
    scanCMD add all methods starting with 'cmd_' as a commands

sendTrackSpeed (vRA, vDEC)

signal_handler (signal, frame)
    Capture Ctrl+C key

zmqQueue (*args, **kwargs)

```

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":'##'}

API

To mimic a tty serial port: *socat TCP:localhost:6000,reuseaddr pty,link=/tmp/lx200*

`termiteOS.nodes.tcpxproxy.runtcpxproxy (name, port, parent_host, parent_port, params)`
ENTRYPOINT calling this function start the node

`class termiteOS.nodes.tcpxproxy.tcpxproxy (name, port, parent_host, parent_port, params)`

HasChildren ()

Return **True** if the node has childrens. **False** otherwise

addCMDs (CMDs)

add commands explicitly

clientthread (conn, parent_host, parent_port)

Function for handling connections. This will be used to create threads

cmd (cmd)

Execute the cmd command

Parameters **cmd** – string contain the command with his parameters

Returns A message containing the answer

cmd_help (arg)

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

cmd_ls (arg)

list the commands

cmd_nodes (arg)

list the children nodes:

Returns a python list with all the children names

cmd_ping (arg)

cmd_tree (arg)

Print all node and children available commands

cmddummy (arg)

Default cmd to execute when not knowed cmd match. Do nothing

deregister (arg)

Close the zmq socket and deleted node from the children list

end (arg='')

Close all and exit

exenodeCmd (arg)

Execute the command in the children

heartbeat (arg)

heartbeat Parent part

nodeheartbeat (*args, **kwargs)

```

recv_end (conn)
    Parse cmd lines

register (arg='')
    Call to parent registrar

registrar (arg)
    Registrar parent part

run ()
    Main loop. Dispatch incoming messages

scanCMDs ()
    scanCMD add all methods starting with 'cmd_' as a commands

signal_handler (signal, frame)
    Capture Ctrl+C key

startserver (port)
    Function to open the TCP incoming port

zmqQueue (*args, **kwargs)

```

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

```

API

ENGINE

```
termiteOS.nodes.telescope.runtelescope (name, port, parent_host='', parent_port=False)
```

ENTRYPOINT calling this fuction start the node

```
class termiteOS.nodes.telescope.telescope (name, port, parent_host, parent_port)
```

HasChildren ()

Return **True** if the node has childrens. **False** otherwise

addCMDs (*CMDs*)

add commands explicately

altAz_of (*ra, dec*)

cmd (*cmd*)

Execute the cmd command

Parameters **cmd** – string contain the command with his parameters

Returns A message containing the answer

cmd_ack (*arg*)

cmd_align2target (*arg*)

cmd_firmware_date (*arg*)

cmd_firmware_ver (*arg*)

cmd_getLocalDate (*arg*)

cmd_getLocalTime (*arg*)

cmd_getSideralTime (*arg*)

cmd_getTargetDEC (*arg*)

cmd_getTargetRA (*arg*)

cmd_getTelescopeDEC (*arg*)

cmd_getTelescopeRA (*arg*)

cmd_help (*arg*)

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

cmd_info (*arg*)

cmd_ls (*arg*)

list the commands

cmd_nodes (*arg*)

list the children nodes:

Returns a python list with all the children names

cmd_ping (*arg*)

cmd_pulseE (*arg*)

cmd_pulseN (*arg*)

cmd_pulseS (*arg*)

cmd_pulseW (*arg*)

cmd_setMaxSlewRate (*arg*)

cmd_setTargetDEC (*arg*)

cmd_setTargetRA (*arg*)

cmd_slew (*arg*)

cmd_slewRate (*arg*)

cmd_stopSlew (*arg*)

cmd_tree (*arg*)

Print all node and children availabled commands

cmddummy (*arg*)
Default cmd to execute when not knowed cmd match. Do nothing

deregister (*arg*)
Close the zmq socket and deleted node from the children list

end (*arg*= '')

exenodeCmd (*arg*)
Execute the command in the children

getDEC (*arg*)

getGear (*arg*)

getObserver (*arg*)

getRA (*arg*)

heartbeat (*arg*)
heartbeat Parent part

hourAngle (*ra*)

nodeheartbeat (**args, **kwargs*)

observerInit ()

register (*arg*= '')
Call to parent registrar

registrar (*arg*)
Registrar parent part

run ()

scanCMDs ()
scanCMD add all methods starting with '**cmd_**' as a commands

setTrackSpeed (*arg*)

signal_handler (*signal, frame*)
Capture Ctrl+C key

track ()

values (*arg*)

zmqQueue (**args, **kwargs*)

Drivers

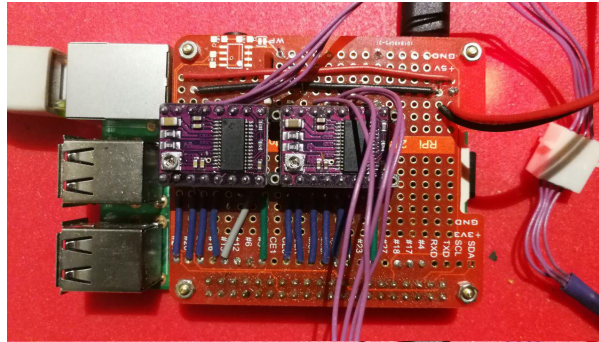
Drivers interact with the real hardware and normaly are used in the **node** code.

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 (*resetpoint*)

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 position (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, 12
- miteJoy command line option, 14
- mitetcp proxy command line option, 15
- miteTelescope command line option, 17
- miteTLE tracker command line option, 10

–params <params>

- mitetcp proxy command line option, 16

–parent_host <parent_host>

- miteHub command line option, 12
- miteJoy command line option, 14
- mitetcp proxy command line option, 15
- miteTelescope command line option, 17
- miteTLE tracker command line option, 10

–parent_port <parent_port>

- miteHub command line option, 12
- miteJoy command line option, 14
- mitetcp proxy command line option, 16
- miteTelescope command line option, 17
- miteTLE tracker command line option, 10

–port <port>

- miteHub command line option, 12
- miteJoy command line option, 14
- mitetcp proxy command line option, 15
- miteTelescope command line option, 17
- miteTLE tracker command line option, 10

A

- addCMDs() (termiteOS.nodes.hub.hub method), 13
- addCMDs() (termiteOS.nodes.joystick.stick method), 14
- addCMDs() (termiteOS.nodes.tcp proxy.tcp proxy method), 16
- addCMDs() (termiteOS.nodes.telescope.telescope method), 18
- addCMDs() (termiteOS.nodes.TLE tracker.TLE tracker method), 11
- altAz_of() (termiteOS.nodes.telescope.telescope method), 18

C

- circle() (termiteOS.nodes.TLE tracker.TLE tracker method), 11
- clientthread() (termiteOS.nodes.tcp proxy.tcp proxy method), 16

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

clutch() (termiteOS.drivers.rpi.rpiSpeedPWM.rpiSpeedPWM method), 21

cmd() (termiteOS.nodes.hub.hub method), 13

cmd() (termiteOS.nodes.joystick.stick method), 14

cmd() (termiteOS.nodes.tcp proxy.tcp proxy method), 16

cmd() (termiteOS.nodes.telescope.telescope method), 18

cmd() (termiteOS.nodes.TLE tracker.TLE tracker method), 11

cmd_ack() (termiteOS.nodes.telescope.telescope method), 18

cmd_align2target() (termiteOS.nodes.telescope.telescope method), 18

cmd_firmware_date() (termiteOS.nodes.telescope.telescope method), 18

cmd_firmware_ver() (termiteOS.nodes.telescope.telescope method), 18

cmd_follow() (termiteOS.nodes.TLE tracker.TLE tracker method), 11

cmd_getLocalDate() (termiteOS.nodes.telescope.telescope method), 18

cmd_getLocalTime() (termiteOS.nodes.telescope.telescope method), 18

cmd_getSideralTime() (termiteOS.nodes.telescope.telescope method), 18

cmd_getTargetDEC() (termiteOS.nodes.telescope.telescope method), 18

cmd_getTargetRA() (termiteOS.nodes.telescope.telescope method), 18

cmd_getTelescopeDEC() (termiteOS.nodes.telescope.telescope method), 18

cmd_getTelescopeRA() (termiteOS.nodes.telescope.telescope method), 18

cmd_help() (termiteOS.nodes.hub.hub method), 13

`cmd_help()` (termiteOS.nodes.joystick.stick method), 14
`cmd_help()` (termiteOS.nodes.tcpproxy.tcpproxy method), 16
`cmd_help()` (termiteOS.nodes.telescope.telescope method), 18
`cmd_help()` (termiteOS.nodes.TLEtracker.TLEtracker method), 11
`cmd_info()` (termiteOS.nodes.telescope.telescope method), 18
`cmd_ls()` (termiteOS.nodes.hub.hub method), 13
`cmd_ls()` (termiteOS.nodes.joystick.stick method), 14
`cmd_ls()` (termiteOS.nodes.tcpproxy.tcpproxy method), 16
`cmd_ls()` (termiteOS.nodes.telescope.telescope method), 18
`cmd_ls()` (termiteOS.nodes.TLEtracker.TLEtracker method), 11
`cmd_nodes()` (termiteOS.nodes.hub.hub method), 13
`cmd_nodes()` (termiteOS.nodes.joystick.stick method), 14
`cmd_nodes()` (termiteOS.nodes.tcpproxy.tcpproxy method), 16
`cmd_nodes()` (termiteOS.nodes.telescope.telescope method), 18
`cmd_nodes()` (termiteOS.nodes.TLEtracker.TLEtracker method), 11
`cmd_ping()` (termiteOS.nodes.hub.hub method), 13
`cmd_ping()` (termiteOS.nodes.joystick.stick method), 14
`cmd_ping()` (termiteOS.nodes.tcpproxy.tcpproxy method), 16
`cmd_ping()` (termiteOS.nodes.telescope.telescope method), 18
`cmd_ping()` (termiteOS.nodes.TLEtracker.TLEtracker method), 11
`cmd_pulseE()` (termiteOS.nodes.telescope.telescope method), 18
`cmd_pulseN()` (termiteOS.nodes.telescope.telescope method), 18
`cmd_pulseS()` (termiteOS.nodes.telescope.telescope method), 18
`cmd_pulseW()` (termiteOS.nodes.telescope.telescope method), 18
`cmd_setMaxSlewRate()` (termiteOS.nodes.telescope.telescope method), 18
`cmd_setTargetDEC()` (termiteOS.nodes.telescope.telescope method), 18
`cmd_setTargetRA()` (termiteOS.nodes.telescope.telescope method), 18
`cmd_slew()` (termiteOS.nodes.telescope.telescope method), 18
`cmd_slewRate()` (termiteOS.nodes.telescope.telescope method), 18
`cmd_stopSlew()` (termiteOS.nodes.telescope.telescope

method), 18
`cmd_tree()` (termiteOS.nodes.hub.hub method), 13
`cmd_tree()` (termiteOS.nodes.joystick.stick method), 14
`cmd_tree()` (termiteOS.nodes.tcpproxy.tcpproxy method), 16
`cmd_tree()` (termiteOS.nodes.telescope.telescope method), 18
`cmd_tree()` (termiteOS.nodes.TLEtracker.TLEtracker method), 11
`cmddummy()` (termiteOS.nodes.hub.hub method), 13
`cmddummy()` (termiteOS.nodes.joystick.stick method), 15
`cmddummy()` (termiteOS.nodes.tcpproxy.tcpproxy method), 16
`cmddummy()` (termiteOS.nodes.telescope.telescope method), 18
`cmddummy()` (termiteOS.nodes.TLEtracker.TLEtracker method), 11

D

`deregister()` (termiteOS.nodes.hub.hub method), 13
`deregister()` (termiteOS.nodes.joystick.stick method), 15
`deregister()` (termiteOS.nodes.tcpproxy.tcpproxy method), 16
`deregister()` (termiteOS.nodes.telescope.telescope method), 19
`deregister()` (termiteOS.nodes.TLEtracker.TLEtracker method), 11

E

`end()` (termiteOS.nodes.hub.hub method), 13
`end()` (termiteOS.nodes.joystick.stick method), 15
`end()` (termiteOS.nodes.tcpproxy.tcpproxy method), 16
`end()` (termiteOS.nodes.telescope.telescope method), 19
`end()` (termiteOS.nodes.TLEtracker.TLEtracker method), 11
`exenodeCmd()` (termiteOS.nodes.hub.hub method), 13
`exenodeCmd()` (termiteOS.nodes.joystick.stick method), 15
`exenodeCmd()` (termiteOS.nodes.tcpproxy.tcpproxy method), 16
`exenodeCmd()` (termiteOS.nodes.telescope.telescope method), 19
`exenodeCmd()` (termiteOS.nodes.TLEtracker.TLEtracker method), 11

F

`fault()` (termiteOS.drivers.rpi.rpiDRV8825Hat.rpiDRV8825Hat method), 20
`fault()` (termiteOS.drivers.rpi.rpiSpeedPWM.rpiSpeedPWM method), 21

G

`gearInit()` (termiteOS.nodes.TLEtracker.TLEtracker method), 11

getDEC() (termiteOS.nodes.telescope.telescope method), 19
 getGear() (termiteOS.nodes.telescope.telescope method), 19
 getObserver() (termiteOS.nodes.telescope.telescope method), 19
 getRA() (termiteOS.nodes.telescope.telescope method), 19
 goto() (termiteOS.drivers.rpi.rpiSpeedPWM.rpiSpeedPWM method), 21
 gotoEnd (termiteOS.drivers.rpi.rpiSpeedPWM.rpiSpeedPWM attribute), 21

H

HasChildren() (termiteOS.nodes.hub.hub method), 13
 HasChildren() (termiteOS.nodes.joystick.stick method), 14
 HasChildren() (termiteOS.nodes.tcproxy.tcproxy method), 16
 HasChildren() (termiteOS.nodes.telescope.telescope method), 17
 HasChildren() (termiteOS.nodes.TLEtracker.TLEtracker method), 11
 heartbeat() (termiteOS.nodes.hub.hub method), 13
 heartbeat() (termiteOS.nodes.joystick.stick method), 15
 heartbeat() (termiteOS.nodes.tcproxy.tcproxy method), 16
 heartbeat() (termiteOS.nodes.telescope.telescope method), 19
 heartbeat() (termiteOS.nodes.TLEtracker.TLEtracker method), 11
 hourAngle() (termiteOS.nodes.telescope.telescope method), 19
 hub (class in termiteOS.nodes.hub), 13

I

isStopped (termiteOS.drivers.rpi.rpiSpeedPWM.rpiSpeedPWM attribute), 21

L

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

M

miteHub command line option
 -name <name>, 12
 -parent_host <parent_host>, 12
 -parent_port <parent_port>, 12
 -port <port>, 12
 miteJoy command line option
 -name <name>, 14
 -parent_host <parent_host>, 14
 -parent_port <parent_port>, 14
 -port <port>, 14
 miteLaunch command line option
 YAMFILE, 10
 mitetcp proxy command line option

-name <name>, 15
 -params <params>, 16
 -parent_host <parent_host>, 15
 -parent_port <parent_port>, 16
 -port <port>, 15

miteTelescope command line option

-name <name>, 17
 -parent_host <parent_host>, 17
 -parent_port <parent_port>, 17
 -port <port>, 17

miteTLEtracker 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

N

nodeheartbeat() (termiteOS.nodes.hub.hub method), 13
 nodeheartbeat() (termiteOS.nodes.joystick.stick method), 15
 nodeheartbeat() (termiteOS.nodes.tcproxy.tcproxy method), 16
 nodeheartbeat() (termiteOS.nodes.telescope.telescope method), 19
 nodeheartbeat() (termiteOS.nodes.TLEtracker.TLEtracker method), 11

O

observerInit() (termiteOS.nodes.telescope.telescope method), 19
 observerInit() (termiteOS.nodes.TLEtracker.TLEtracker method), 11

R

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

R

rampUp() (termiteOS.drivers.rpi.rpiSpeedPWM.rpiSpeedPWM method), 22
 recv_end() (termiteOS.nodes.tcproxy.tcproxy method), 16
 register() (termiteOS.nodes.hub.hub method), 13
 register() (termiteOS.nodes.joystick.stick method), 15
 register() (termiteOS.nodes.tcproxy.tcproxy method), 17
 register() (termiteOS.nodes.telescope.telescope method), 19
 register() (termiteOS.nodes.TLEtracker.TLEtracker method), 11
 registrar() (termiteOS.nodes.hub.hub method), 13
 registrar() (termiteOS.nodes.joystick.stick method), 15
 registrar() (termiteOS.nodes.tcproxy.tcproxy method), 17

registrar() (termiteOS.nodes.telescope.telescope method), 19

registrar() (termiteOS.nodes.TLEtracker.TLEtracker method), 12

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() (termiteOS.nodes.hub.hub method), 13

run() (termiteOS.nodes.joystick.stick method), 15

run() (termiteOS.nodes.tcproxy.tcproxy method), 17

run() (termiteOS.nodes.telescope.telescope method), 19

run() (termiteOS.nodes.TLEtracker.TLEtracker method), 12

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

runhub() (in module termiteOS.nodes.hub), 13

runjoystick() (in module termiteOS.nodes.joystick), 14

runtcproxy() (in module termiteOS.nodes.tcproxy), 16

runtelescope() (in module termiteOS.nodes.telescope), 17

runTLEtracker() (in module termiteOS.nodes.TLEtracker), 12

S

satPosition() (termiteOS.nodes.TLEtracker.TLEtracker method), 12

satSpeed() (termiteOS.nodes.TLEtracker.TLEtracker method), 12

scanCMDs() (termiteOS.nodes.hub.hub method), 13

scanCMDs() (termiteOS.nodes.joystick.stick method), 15

scanCMDs() (termiteOS.nodes.tcproxy.tcproxy method), 17

scanCMDs() (termiteOS.nodes.telescope.telescope method), 19

scanCMDs() (termiteOS.nodes.TLEtracker.TLEtracker method), 12

sendSlew() (termiteOS.nodes.TLEtracker.TLEtracker method), 12

sendTrackSpeed() (termiteOS.nodes.joystick.stick method), 15

sendTrackSpeed() (termiteOS.nodes.TLEtracker.TLEtracker method), 12

set_dir() (termiteOS.drivers.rpi.rpiDRV825Hat.rpiDRV825Hat method), 20

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

set_microsteps() (termiteOS.drivers.rpi.rpiDRV825Hat.rpiDRV825Hat method), 20

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

set_point() (termiteOS.drivers.rpi.rpiSpeedPWM.rpiSpeedPWM method), 21

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

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

setTrackSpeed() (termiteOS.nodes.telescope.telescope method), 19

signal_handler() (termiteOS.nodes.hub.hub method), 13

signal_handler() (termiteOS.nodes.joystick.stick method), 15

signal_handler() (termiteOS.nodes.tcproxy.tcproxy method), 17

signal_handler() (termiteOS.nodes.telescope.telescope method), 19

signal_handler() (termiteOS.nodes.TLEtracker.TLEtracker method), 12

sleep() (termiteOS.drivers.rpi.rpiDRV825Hat.rpiDRV825Hat method), 20

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

startserver() (termiteOS.nodes.tcproxy.tcproxy method), 17

stepcounter() (termiteOS.drivers.rpi.rpiDRV825Hat.rpiDRV825Hat method), 20

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

stick (class in termiteOS.nodes.joystick), 14

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

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

sync() (termiteOS.drivers.rpi.rpiDRV825Hat.rpiDRV825Hat method), 20

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

T

tcproxy (class in termiteOS.nodes.tcproxy), 16

telescope (class in termiteOS.nodes.telescope), 17

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

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

termiteOS.launch (module), 10

termiteOS.nodes.hub (module), 13

termiteOS.nodes.joystick (module), 14

termiteOS.nodes.tcproxy (module), 16

termiteOS.nodes.telescope (module), 17

termiteOS.nodes.TLEtracker (module), 11

termiteOS.drivers.rpi.rpiDRV825Hat.rpiDRV825Hat method), 20

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

threaded() (in module termiteOS.drivers.rpi.rpiSpeedPWM), [22](#)
TLEtracker (class in termiteOS.nodes.TLEtracker), [11](#)
track() (termiteOS.nodes.telescope.telescope method), [19](#)
trackSatellite() (termiteOS.nodes.TLEtracker.TLEtracker method), [12](#)
trackSpeed() (termiteOS.drivers.rpi.rpiSpeedPWM.rpiSpeedPWM method), [22](#)

V

values() (termiteOS.nodes.telescope.telescope method), [19](#)

Y

YAMLFIELD
miteLaunch command line option, [10](#)

Z

zmqQueue() (termiteOS.nodes.hub.hub method), [13](#)
zmqQueue() (termiteOS.nodes.joystick.stick method), [15](#)
zmqQueue() (termiteOS.nodes.tcproxy.tcproxy method), [17](#)
zmqQueue() (termiteOS.nodes.telescope.telescope method), [19](#)
zmqQueue() (termiteOS.nodes.TLEtracker.TLEtracker method), [12](#)