HFOS Documentation

Release 1.2.1

Hackerfleet Contributors

Oct 02, 2018

Contents

Version 1.2.1

Release 1.2.1

Date Oct 02, 2018

CHAPTER 1

About

1.1 Hackerfleet

1.1.1 Who we are

The Hackerfleet is a research & development venture founded by some friends who decided to revolutionize the maritime technology sector.

We develop opensource hardware and software for unmanned and manned vessels on all waters, we do this publicly and transparent, our repositorys are open for you.

1.1.2 Our goals

One of our primary goals is to establish a better communication network between compatible hardware, for automatic data exchange between ships.

We want to aggregate all the information that is currently thrown away on shipbridges all over the world and make the best free map of the ocean.

This vast ressource of currently nearly unused data will also help scientists understand our oceans better.

1.1.3 Timeline

- 2011 Founding -> CCC Camp MS 0x00
- 2012 Hackathon for Android App 'Social Bearing'
- 2013 Mariner's code: Computer hackers conquering the high seas
- 2014 EuroPython Hackathons
- 2015 Oh, camp again! We did some crowdsourced management

And now, we're here!

You can check all this out on the intertubes. Youtube, CNN, etc. Just search for 'Hackerfleet' - hmm.. succinct name, eh?

1.1.4 Founders

- Heiko ‘riot’ Weinen (riot@c-base.org, @__r107__, ri0t@github)
- Johannes ‘ijon’ Rundfeldt (ijon@c-base.org, @aegrereminiscen, ij0n@github)

Meet us at c-base, the spacestation below Berlin Mitte!

1.1.5 Communication

- github.com/Hackerfleet
- Waffle.io/hackerfleet/hfos
- Twitter.com/hackerfleet
- Facebook/hackerfleet
- hackerfleet.soup.io
- Also on G+
- irc #hackerfleet on freenode

Note: Please be patient when using IRC, responses might take a few hours!

1.2 HFOS

1.2.1 About HFOS

The Hackerfleet Operating System (Short HFOS) is being developed specifically to target a handful of properties and challenges, that are unique to the projected use of the system:

- Locally offline and undisruptable operation (True Internet!)
- Extremely low energy profile
- Must work on embedded systems with low memory, storage and computing capacity
- Realtime handling and federation of incoming and outgoing data
- Many, many different bus and sensor systems as well as configurations
- Clients should not be limited in any way

To master all these challenges, a rather radical approach was chosen after evaluation of most of the currently available frameworks and libraries.

1.2.2 HFOS System overview

Architecture

The system consists of two parts: 1. A backend written in Python. It handles communications, data handling and other general services provided by independent modules. 2. To communicate with users, a HTML5 based Frontend is deployed to most modern web browser capable clients.

Meshed Operation

The cloud/server-less mesh operation enables local independence and adaptability regarding network environments.

No platform specifics

It also eliminates the need to write platform specific applications (e.g. native Android or other mobile platform applications)

1.2.3 What's new here?

At first glance, HFOS looks like just another web application platform.

In contrast to most available systems though, HFOS works using a component based frontend and backend architecture.

This enables every installation to install, activate and use only the modules relevant for the local group of users.

Also, communication between clients and the backend has been streamlined and minimized by relying on Websockets.

CHAPTER 2

Sailor's Manual

2.1 Getting Started

2.1.1 Quick Start Guide

Install script

The script currently only supports Debian based systems.

Note: Feel free to contribute installation steps for other distros - that is mostly adapting the package manager and package names

To use the install script, get the source code (see *Getting the source*) if you don't have it already, then invoke the script with root permissions:

```
$ sudo ./install
```

If you run into trouble or get any unexpected errors, *try the complex installation procedure*.

Docker

We're providing a Docker image for installation.

The command to get the current testing release is:

```
$ docker run -i -t hackerfleet/hfos hfos_launcher.py
```

Planned Installations

- We're planning to offer ready-made SD card images for various embedded systems.
- A custom NixOS system is planned as well.

2.1.2 Requirements and Dependencies

Backend

HFOS' backend has a few dependencies:

- Python: >= 3.3 (or possibly pypy >= 2.0)
- Database: MongoDb

Note: We're phasing out Python 2.7 support.

A few more dependencies like nginx, and some python packages provided per distribution are recommended, but not strictly necessary.

The HFOS Python package additionally installs a few pure Python libraries:

- Circuits
- Click and a few supporting packages
- PyMongo
- PyOpenSSL
- PyStache
- JSONSchema
- DPath
- DeepDiff

  Supported Platforms Linux

  Supported Python Versions (2.7), 3.3, 3.4, 3.5, 3.6

Frontend

The frontend is built with

- node
- npm

and others. The detailed list can be found in frontend/package.json after pulling the frontend git submodule.

Todo: Link backend deps

2.1.3 Downloading

Latest Stable Release

By design, there is currently no stable release planned.

The latest stable releases (if there should ever be one) could be downloaded from the Releases page (*specifically the Tags tab*).

Latest Development Source Code

We use Git for source control and code sharing.

The latest development branch can be cloned using the following command:

```
$ git clone https://github.com/hackerfleet/hfos.git
```

For further instructions on how to use Git, please refer to the Git Website.

2.1.4 Installing

First of all: The manual installation procedure is rather complex right now.

We've *simplified the process by supplying an install script*, but if you encounter any trouble/problems, checkout these detailed installation steps.

If you still can't get it to install, *contact us via irc or email* and we'll happily try to help you get your installation running.

This is very important for us, since the system has not yet been deployed very often and we're not yet aware of all of the pitfalls and traps on that route.

We encourage you to use Python >= 3.5 for HFOS, but the system is built (and checked against) 2.7, too.

Warning: **HFOS is not compatible with Python 3.2!**

Preparation

These instructions are for Debian or Ubuntu based systems. Installation on other distributions is possible.

Before doing anything with HFOS, be sure you have all the dependencies installed via your distribution's package manager.

For Debian Unstable use this:

```
$ sudo apt-get install nginx mongodb python3.5 python3-pip python3-grib \
                       python3-bson python3-pymongo python3-serial
```

If you want (and can, depending on your platform/distribution), install the mongo and bson extensions for speedups:

```
$ sudo apt-get install python3-pymongo-ext python3-bson-ext
```

The system will need to get a bunch of more dependencies via npm to set up the frontend, so install npm and if necessary the nodejs-legacy-symlink package:

```
$ sudo apt-get install npm nodejs
$ sudo npm install npm@4.2.0 -g
```

If you want to install the full development dependencies to write documentation as well, you need to install the enchant package:

```
$ sudo apt-get install enchant
```

In case you want to use raster (or in future: vector) charts in HFOS' map module, you'll need to install libgdal and its binaries:

```
$ sudo apt-get install gdal-bin python-gdal
```

Note, that it is necessary to install python-gdal 2.7 - not the python3 variant, as the scripts are not included in that.

Getting the source

To initially obtain the development source code if you don't have it already, use git thus:

```
$ mkdir ~/src
$ cd ~/src
$ git clone https://github.com/hackerfleet/hfos
$ cd hfos
$ git submodule init
$ git submodule update
$ git pull
$ cd frontend
$ git pull
```

Backend

The manage tool usually can install everything you need. It starts by adding a new system user for HFOS and generating a (currently only self signed) certificate.

The process also involves installing the supplied modules, getting the frontend dependencies, building and installing the documentation, etc.

It also creates two folders in /var (lib/hfos and cache/hfos) for hfos' tile- cache and other stuff as well as install basic default provisions into the database.

Finally, it installs and activates a systemd and nginx service script to launch HFOS on bootup and make it available to users.

```
$ virtualenv -p /usr/bin/python3.5 --system-site-packages venv
$ source venv/bin/activate
$ pip install -Ur requirements.txt
$ python setup.py develop
$ sudo venv/bin/python hfos_manage.py install all
```

If you want to develop (documentation) as well, you'll need to use the *requirements-dev.txt* instead of the normal one.

If you want to manually start HFOS, invoke the launcher thus:

```
$ sudo ./venv/bin/python hfos_launcher.py
```

Running the launcher as root to be able to open ports below 1024 should be safe, as it drops its root privileges, unless you specify –insecure, which is strongly discouraged and only meant for development purposes. The default is to use port 8055 and relay that with the supplied nginx site definition

Documentation

The documentation is available online on ReadTheDocs.org. If you wish to build and install the included documentation for offline use, run these commands:

```
$ sudo ./venv/bin/python hfos_manage.py install docs
```

This installs all necessary documentation tools and copies the files to the expected HFOS web data folder.

You can also build the PDF file (and various other formats) by using the Makefile inside the docs directory.

```
$ cd docs
$ make pdf
```

Just running make without arguments gives you a list of the other available documentation formats.

Installing from a Source Package

If you have downloaded a source archive, this applies to you.

```
$ python setup.py install
```

For other installation options see:

```
$ python setup.py --help install
```

Installing from the Development Repository

If you have cloned the source code repository, this applies to you.

If you have cloned the development repository, it is recommended that you use setuptools and use the following command:

```
$ python setup.py develop
```

This will allow you to regularly update your copy of the hfos development repository by simply performing the following in the hfos working directory:

```
$ git pull -u
$ cd frontend
$ git pull -u
```

Note: You do not need to reinstall if you have installed with setuptools via the hfos repository and used setuptools to install in "develop" mode.

Windows & OS X installation notes

These instructions are WiP. The easiest way to get HFOS on Win7 or newer is to install and user Docker or a virtual machine

To install on Windows, you'll need to install these packages first:

- Python 3.5 https://www.python.org/downloads/windows/
- MongoDB https://www.mongodb.org/downloads#production
- pymongo
- numpy

Platform specific

There are some collected instructions for various hardware platforms:

Raspberry Pi

Swap

Since this machine doesn't have much RAM, don't forget to add a swap partition or file.

2.1.5 Command Line Tools

HFOS contains a few CLI scripts:

HFOS Launcher

hfos_launcher

Bootstrap basics, assemble graph and hand over control to the Core component

```
hfos_launcher [OPTIONS]
```

Options

-p, --port <port>
Define port for server

--host <host>
Define hostname for server

-c, --cert, --certificate <cert>
Certificate file path

--dbhost <dbhost>
Define hostname for database server

--dbname <dbname>
Define name of database (default: hfos)

--profile
Enable profiler

--opengui
Launch webbrowser for GUI inspection after startup

--drawgraph
Draw a snapshot of the component graph after construction

-q, --quiet
Suppress console output

--log <log>
Define console log level (0-100)

--logfileverbosity <logfileverbosity>
Define file log level (0-100)

--logfilepath <logfilepath>
Logfile path

--dolog
Write to logfile

--livelog
Log to in memory structure as well

--debug
Run circuits debugger

--dev
Run development server

--instance <instance>
Define name of instance

--insecure
Keep privileges - INSECURE

--norun
Only assemble system, do not run

-b, --blacklist <blacklist>
Blacklist a component

Management Tool

Todo: Simplify the installation documents by stringing them together, so the other setup documents are more visible

Todo: Add setup section

Todo: Add crew setup

Todo: Add vessel setup

Todo: Add generic preferences setup

2.2 HFOS User's Manual

Welcome to the HFOS Users Manual! This part of the documentation explains how to work with HFOS and use the core modules.

2.2.1 Detailed Feature Overview

CHAPTER 3

Developer Documentation

3.1 How to help the project?

Glad to see you're interested in helping out the project!

Generally, you can ping riot if you want to help out and don't exactly know where to start.

Here, we list a few possible opportunities where you can help us and become part of the driving community:

3.1.1 Communication

People need to be more aware of this project as it may be of great value to them. If you're interested in spreading the word and getting people involved, you're very welcome to do so. Again, please ping riot to get crucial info on how to do so.

3.1.2 Testing

There are various degrees to which you can test the project:

- Check the installation processes if they actually work on your platform and everything installs smoothly
- Test-drive your installation or the demo instance (Currently offline for maintenance)
- Build and extend parts of the automatic testing infrastructure
- Optimize and extend the continuous integration infrastructure

3.1.3 User Experience

We'd value your input on some very important user experience questions:

- Is the current design logical and does it allow for a smooth user experience?
- Check the supplied modules and the framework itself for consistency and good UX practices
- Develop further use cases and user stories to spark new modules

3.1.4 Documentation

A lot of documentation is still missing. If you're interested in writing documentation, you should be familiar with the two core tools we use for generating our documentation:

- reStructured Text formatting
- Sphinx

We still need a lot of module, core framework and source code documentation, so there's ample opportunities in this field.

3.1.5 Translations

Most of (if not all) parts of the project can be translated and are waiting for your help. You can use Transifex to translate all the strings we have or work with your favourite PO Editor. Have a look at *Translating HFOS* for more details.

3.2 Developer Guidelines

This is the rather dry material for new software developers:

3.2.1 Development Introduction

Here's how we do things in HFOS. . .

Communication

- IRC Channel on the FreeNode IRC Network
- Issue Tracker

Note: If you are familiar with IRC and use your own IRC Client then connect to the FreeNode Network and `/join #hackerfleet`.

Standards

We use the following coding standard:

- PEP-008

We also lint our codebase with the following tools:

- pyflakes
- pep8
- mccabe

Please ensure your Development IDE or Editor has the above linters and checkers in place and enabled.

Alternatively you can use the following command line tool:

- flake8

Tools

We use the following tools to develop HFOS and share code:

- **Code Sharing:** Git
- **Code Hosting and Bug Reporting:** GitHub
- **Issue Tracker:** Issue Tracker
- **Documentation Hosting:** Read the Docs
- **Package Hosting:** Python Package Index (PyPi)
- **Docker Hub Automated Builds:** Dockerhub
- **Continuous Integration:** Travis CI
- **Code Quality:** Landscape
- **Frontend Testing:** Browserstack
- **Translations:** Transifex

We strongly suggest familiarizing with all of them, to make sure you understand our CI.

Big thanks to all of these magnificent and free-for-opensource services!

3.2.2 Contributing to HFOS

Here's how you can contribute to HFOS

Submitting Bug Reports

We welcome all bug reports. We do however prefer bug reports in a clear and concise form with repeatable steps. One of the best ways you can report a bug to us is by writing a unit test (//similar to the ones in our tests//) so that we can verify the bug, fix it and commit the fix along with the test.

To submit a bug report, please Create an Issue

Writing new tests

We're not perfect, and we're still writing more tests to ensure quality code. If you'd like to help, please Fork HFOS, write more tests that cover more of our code base and submit a Pull Request. Many Thanks!

Adding New Features

If you'd like to see a new feature added to HFOS, then we'd like to hear about it~ We would like to see some discussion around any new features as well as valid use-cases. To start the discussions off, please either:

- Chat with us on #hackerfleet on the FreeNode IRC Network

  or

- Create an Issue

3.2.3 Setting up a HFOS Development Environment

This is the recommended way to setup a development enviornment for developing the backend and frontend of HFOS .

Note: This document *assumes* you already have a working Python environment with a minimum Python version of 2.7 as well as mongodb and pip already installed.

Prerequisites

It is highly recommended that you install and use virtualenv for all your Python development and production deployments (*not just HFOS*).

It is also convenient to install and use the accompanying shell scripts and tools virtualenvwrapper which adds a nice set of workflows and functions useful for both development and deployments.

```
$ pip install -U virtualenvwrapper
$ source $(which virtualenvwrapper.sh)
```

Note: You should put `source $(which virtualenvwrapper.sh)` in either your `$HOME/.bashrc` or `$HOME/.profile` depending on how you login and interact with your terminals.

In addition to the above recommendations you must also have a Git client installed and ready to use as well as your Editor/IDE of choice ready to use.

Getting Started

1. Fork HFOS (*if you haven't done so already*)
2. Clone your forked repository using Git
3. Create a new virtual environment using virtualenvwrapper
4. Install the Development Requirements
5. Install HFOS in "develop" mode

And you're done!

Example:

```
$ git clone git@github.com:yourgithubaccount/hfos.git
$ cd hfos
$ mkvirtualenv hfos
$ pip install -r requirements-dev.txt
$ python setup.py develop
```

3.2.4 Development Processes

We document all our internal development processes here so you know exactly how we work and what to expect. If you find any issues or problems, please let us know!

Software Development Life Cycle (SDLC)

We employ the use of the SCRUM Agile Process and use our Issue Tracker to track features, bugs, chores and releases. If you wish to contribute to HFOS, please familiarize yourself with SCRUM and GitHub's Issue Tracker.

Bug Reports

- New Bug Reports are submitted via: https://github.com/hackerfleet/hfos/issues
- Confirmation and Discussion of all New Bug Reports.
- Once confirmed, a new Bug is raised in our Issue Tracker
- An appropriate milestone will be set (*depending on current milestone's schedule and resources*)
- A unit test developed that demonstrates the bug's failure.
- A fix developed that passes the unit test and breaks no others.
- A New Pull Request created with the fix.

  This should contain: - A new or modified unit test. - A patch that fixes the bug ensuring all unit tests pass. - The Change Log updated. - Appropriate documentation updated.
- The Pull Request is reviewed and approved by at least two other developers.

Feature Requests

- New Feature Requests are submitted via: https://github.com/hackerfleet/hfos/issues
- Confirmation and Discussion of all New Feature Requests.
- Once confirmed, a new Feature is raised in our Issue Tracker
- An appropriate milestone will be set (*depending on current milestone's schedule and resources*)
- A unit test developed that demonstrates the new feature.
- The new feature developed that passes the unit test and breaks no others.
- A New Pull Request created with the fix.

  This must contains: - A new or modified unit test. - A patch that implements the new feature ensuring all unit tests pass. - The Change Log updated. - Appropriate documentation updated.
- The Pull Request is reviewed and approved by at least two other developers.

Writing new Code

- Submit a New Issue
- Write your code.
- Use flake8 to ensure code quality.
- Run the tests:

```
$ tox
```

- Ensure any new or modified code does not break existing unit tests.
- Update any relevant doc strings or documentation.
- Update the Change Log appropriately.
- Submit a New Pull Request.

Running the Tests

To run the tests you will need the following installed:

- tox installed as well as
- pytest-cov
- pytest

All of these can be installed via `pip install -r requirements-dev.txt`.

Please also ensure - if you can - that you you have all supported versions of Python that HFOS supports installed in your local environment.

To run the tests:

```
$ tox
```

3.2.5 Development Standards

We aim for the following development standards:

Cyclomatic Complexity

- Code Complexity shall not exceed `10`

  See: Limiting Cyclomatic Complexity

Coding Style

- Code shall confirm to the PEP8 Style Guide.

Note: This includes the 79 character limit!

- Doc Strings shall confirm to the PEP257 Convention.

Note: Arguments, Keyword Arguments, Return and Exceptions must be documented with the appropriate Sphinx Python Domain.

Revision History

- Commits shall be small tangible pieces of work. - Each commit must be concise and manageable. - Large changes are to be done over smaller commits.
- There shall be no commit squashing.
- Rebase your changes as often as you can.

Unit Tests

- Every new feature and bug fix must be accompanied with a unit test. (*The only exception to this are minor trivial changes*).

3.2.6 Translating HFOS

Since 2018, we have all parts (Backend, Frontend, Modules) prepared for translations.

To translate HFOS, you can use Transifex or any PO editor of your choice.

3.3 Recent Changes

- **Moved some WiP stuff out, added new stuff** by *ri0t* at *2018-10-02 20:32:22*
- **Order of arguments fixed** by *ri0t* at *2018-10-02 20:29:31*
- **Logging optimized** by *ri0t* at *2018-10-02 20:29:12*
- **Long argument names should come first** by *ri0t* at *2018-10-02 20:18:39*
- **Frontend ref updated** by *ri0t* at *2018-10-02 20:04:55*
- **Todo list now working** by *ri0t* at *2018-10-02 20:04:32*
- **Sidebar, state persistence and other features added** by *ri0t* at *2018-10-02 20:04:27*
- **Resizer Fixes** by *ri0t* at *2018-10-02 20:04:22*
- **Multi-field inserts in forms** by *ri0t* at *2018-10-02 20:03:50*
- **Minor fixes** by *ri0t* at *2018-10-02 20:03:43*

3.4 Road Map

We manage our roadmap via milestones on our github issuetracker.

3.5 Contributors

The following users and developers have contributed to HFOS:

- Heiko 'riot' Weinen (Primary maintainer)
- Johannes 'ijon' Rundfeldt
- Martin Ling

HFOS proudly uses the circuits framework. circuits was originally designed, written and primarily maintained by James Mills (http://prologic.shortcircuit.net.au/).

Anyone not listed here, ping us. We appreciate any and all contributions to HFOS and other Hackerfleet components.

3.6 Supporters

- **Repository and issue tracker hosting:** Github
- **Free OSS license of IntelliJ IDEA Ultimate:** Jetbrains
- **Free OSS cross platform/browser user interface testing:** Browserstack
- **Initial project conception phase funding:** Kenny Bentley
- **Hosting and nix expertise:** Lassulus

Todo: Remove/Merge original list and asset docs to this document

3.7 Frequently Asked Questions

3.7.1 General

… What is HFOS? HFOS is an opensource navigation and communication platform.

… What platforms does HFOS support? We currently test on Debian, various flavours of Python (3.3, 3.4, 3.5, 3.6, pypy) It'll probably run on various other platforms as well. E.g. we've made good experiences with Arch Linux.

Got more questions?

- Meet us and chat with us online on the #hackerfleet IRC Channel

Note: Please be patient when using IRC, responses might take a few hours!

3.8 Glossary

GPS a navigational system involving satellites and computers that can determine the latitude and longitude of a receiver on Earth by computing the time difference for signals from different satellites to reach the receiver [syn: {Global Positioning System}, {GPS}]

NMEA National Marine Electronics Association [protocol] (org., USA, GPS), http://www.nmea.org

Radar measuring instrument in which the echo of a pulse of microwave radiation is used to detect and locate distant objects [syn: {radar}, {microwave radar}, {radio detection and ranging}, {radiolocation}]

VCS Version Control System, what you use for versioning your source code

3.9 Documentation TODO

3.9.1 Global TODO

- Clean up docstrings
- Shorten Index
- Split up Index?
- Shorten Index title lengths
- Import nautical glossary
- Multilang

3.9.2 Local TODO

Todo: Remove/Merge original list and asset docs to this document

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/hfos/checkouts/latest/docs/source/contributors.rs line 32.)

Todo: Simplify the installation documents by stringing them together, so the other setup documents are more visible

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/hfos/checkouts/latest/docs/source/start/index.rst, line 13.)

Todo: Add setup section

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/hfos/checkouts/latest/docs/source/start/index.rst, line 14.)

Todo: Add crew setup

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/hfos/checkouts/latest/docs/source/start/index.rst, line 15.)

Todo: Add vessel setup

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/hfos/checkouts/latest/docs/source/start/index.rst, line 16.)

Todo: Add generic preferences setup

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/hfos/checkouts/latest/docs/source/start/index.rst, line 17.)

Todo: Link backend deps

(The original entry is located in /home/docs/checkouts/readthedocs.org/user_builds/hfos/checkouts/latest/docs/source/start/requireme line 49.)

3.10 Development README Page

3.10.1 HFOS - The Hackerfleet Operating System

A collaborative and modular infrastructure for your data.

- **Geo Information** Use a sophisticated map to annotate and review geographical information
- **Vehicle support** Attach a sailyacht, your camper or pack one in your backpack
- **Project planning** Issue tracking for collaborative teams
- **Modular** Expandable with integrated modules, build your own
- **Cloud independent** Run nodes on your own infrastructure

3.10.2 Installation

There is more than one way of installing HFOS, see the quickstart instructions for those.

The simplest way is to use the supplied installation script:

```
sudo ./install
```

The installation produces a lot of output which is automatically piped into 'output.log'. It installs all dependencies then sets up a user account and the system's services. This assumes, that you're not running another web server on port 443 and that your firewall is configured to allow communications on that port. HFOS modules may require additional open ports, to find out about that, read their readme files.

If you run into trouble or get any unexpected errors, contact us or try the complex installation procedure.

3.10.3 Modules

The system is modular, so you can install what you need and leave other things.

A lot of the included modules are still Work in Progress, so help out, if you're interested in a powerful - **cloud independent** - collaboration tool suite.

General modules

These are 'official' Hackerfleet developed modules. If you'd like to contribute your own, ping riot@c-base.org, to get it added to the list.

Name	Description
sails	Web UI, compatible with all modern browsers
automat	Automation for non programmers
alert	User alerting and notification system
calc	Integrated EtherCalc
camera	Camera support
chat	Integrated chat
comms	Communication package
countables	Count arbitrary things
dash	Dashboard information system
enrol	Enrollment (new user) management
equipment	Equipment management
filemanager	File management
garden	Garden automation tools
ldap	LDAP user authorization
library	Library management
mesh	Mesh networking
nodestate	Node wide status system
polls	Tool for lightweight internet voting
project	Project management tools
protocols	Miscellaneous communication protocols
robot	RC remote control unit
shareables	Shared resource blocking tool
switchboard	Virtual switchboard
wiki	Etherpad + Wiki = awesomeness

Many of these are not yet fully usable, so please help out and perhaps take ownership of one (or more) of them!

Navigation (Hackerfleet) modules

We primarily focused on navigation tools, so these are currently the 'more usable' modules. They are far from complete, see the WiP list below.

Obligatory Warning: **Do not use for navigational purposes!** *Always have up to date paper maps and know how to use them!*

Name	Description
anchor	Automatic anchor safety watcher
busrepeater	Tool to repeat navigation data bus frames to other media
logbook	Displaying and manual logging of important (nautical) events
maps	(Offline) moving maps with shareable views/layers
navdata	Navigational data module
nmea	NMEA-0183 Navigation data and AIS bus parser
webguides	Importer for skipperguide.de wiki content into the map

Work in progress

- Full GDAL based vector chart support (Currently only raster charts)
- Dynamic Logbook
- GRIB data (in charts)
- Navigation aides, planning
- Datalog, automated navigational data exchange
- Crew management, more safety tools
- wireless crew network and general communications

3.10.4 Bugs & Discussion

Please research any bugs you find via our Github issue tracker for HFOS and report them, if they're still unknown.

If you want to discuss distributed, opensource (or maritime) technology in general incl. where we're heading, head over to our Github discussion forum . . . which is cleverly disguised as a Github issue tracker.

You can also find us here:

- github.com/Hackerfleet
- reddit
- Twitter
- Facebook
- soup.io
- G+
- irc #hackerfleet on freenode

Note: Please be patient when using IRC, responses might take a few hours!

3.10.5 Contributors

Code

- Heiko 'riot' Weinen riot@c-base.org
- Johannes 'ijon' Rundfeldt ijon@c-base.org
- Martin Ling
- Sascha 'c_ascha' Behrendt c_ascha@c-base.org

Assets

- Fabulous icons by iconmonstr.com and Hackerfleet contributors

Support

- c-base e.V. our home base, the spacestation below Berlin Mitte
- Lassulus for hosting and nix expertise
- Github for hosting our code
- Travis.CI for continuous integration services
- BrowserStack for cross device testing capabilities

3.10.6 License

Copyright (C) 2011-2018 riot <riot@c-base.org> and others.

This program is free software: you can redistribute it and/or modify it under the terms of the GNU Affero General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Affero General Public License for more details.

You should have received a copy of the GNU Affero General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

– :boat: :+1:

CHAPTER 4

Indices and tables

- Index
- modindex
- search
- *Glossary*
- *Recent Changes*
- *Documentation TODO*
- *Development README Page*

Index