BoatPi

A Boat Navigation System Using a Raspberry Pi and OpenCPN

Lukas Galke Melanie Poech

2017/12/24

Contents

1 Boat Navigation System

The present document comprises a guide for building a navigation system for a boat using a Raspberry Pi. We focus on creating a chart plotter by combining the Raspberry Pi with an external display. We make use of the OpenCPN[1] software along with a dedicated GPS module to display the current location on a map.

This guide extends the work of Loschwitz (2016) by a description of a more sustainable installation of the OpenCPN software. The guide by Christoffersen (2016) describes a setup using an on-board GPS module. The author gives helpful insights on the electronic properties of the setup. Considering software, the guide relies on the openplotter platform[2].

[1] https://opencpn.org
[2] http://www.sailoog.com/openplotter

The platform offers multiple features besides the chart plotter itself, such as weather forecasting, a compass, or several interfaces for an automated boat control system. The openplotter platform can also be deployed on a Raspberry Pi. Still, the extra features require respective hardware components. For the present guide, we instead focus on the chart plotting software OpenCPN which only requires a GPS module.

In the following, we first describe the basic setup of the Raspberry Pi as well as the GPS and the chart plotter software. Then, we depict optimization methods to optimize the Raspberry's performance and highlight some boat-specific adaptions which need to be considered independently of this guide.

2 Preparation

Requirements

- Raspberry Pi
- Power Supply
- An SD card with Raspbian (jessie or higher) installed
- GPS Module
- Mouse and Keyboard
- Display (preferably without additional power demands)

We start by setting up the Raspberry Pi. Insert the SD card, attach mouse and keyboard. Follow the manufacturer's instruction to set up the display. Connect the GPS module with the RPi and finally, connect the power supply. After the Raspberry has booted, assert that its software is up-to-date (`apt-get update && apt-get upgrade`).

2.1 Installing the GPS Service

Next, we install the software to communicate with the GPS module, namely `gpsd`. To install the `gpsd` GPS daemon, issue the following command:

```
sudo apt-get install gpsd gpsd-clients python-gps
```

After the `gpsd` package is installed, we need to supply the interface identifier, on which the service should listen. To modify the service's configuration, edit the `/etc/default/gpsd` file. Insert the path for the interface (e.g. `/dev/ttyACM0`) of the GPS module into the `DEVICES` line. The correct interface can be identified by inspecting the output of `ls /dev/`. In the same file, add `-b` to `GPSD_OPTIONS`. Restart the `gpsd` service via `service gpsd restart`. Assert that the GPS service is working properly by issuing `cgps` in a command prompt. In case something went wrong, consult the detailed installation instructions of `gpsd`[3].

[3] http://www.catb.org/gpsd/installation.html

2.2 Setting up OpenCPN

Now, we install the OpenCPN[4] navigation software on the Raspberry Pi:

First, edit the package sources via `sudo nano /etc/apt/sources.list` and append the following line:

```
deb http://ppa.launchpad.net/opencpn/opencpn/ubuntu/ trusty main
```

Make sure to save the file after editing. Then, retrieve the key from the keyserver via

```
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys C865EB40
```

On a command prompt and update the package list via

```
sudo apt-get update
```

Finally, install the `opencpn` software itself by

```
sudo apt-get install opencpn
```

Optional, if desired, install available OpenCPN plugins by append a `*`

```
sudo apt-get install opencpn*
```

Start the new software by entering `opencpn &` on a command prompt or via the graphical user interface. When there are problems with the installation or further information on the usage of OpenCPN, please refer to the OpenCPN User Manual[5].

3 Getting on Board

When deploying the navigation system on the boat, some key aspects need to be considered. Since a boat navigation system is a real-time application, the software should run fluently on the display. The showcased methods to optimize the BoatPi's overall performance also reduce its power demand. The power source must match the demands in terms of both voltage and amperage. Finally, the charts need to be present in a OpenCPN-compatible format.

3.1 Reduce Screen Resolution

When the GPU of the Raspberry Pi is to weak or the power consumption is too high, reduce the screen resolution. Run `sudo raspi-config`, navigate to *Advanced Options* > *Resolution* and set resolution according to your display (720p is recommended). Run `sudo reboot`, to let changes within `raspi-config` take effect.

[4] https://opencpn.org
[5] https://opencpn.org/wiki/dokuwiki/doku.php?id=opencpn:opencpn_user_manual

3.2 Adjust the Framebuffer

To further optimize the performance of the Raspberry Pi, Loschwitz (2016) suggests to set:

```
framebuffer_depth=32
framebuffer_ignore_alpha=1
```

in the `/boot/config.txt` file, before performing a reboot to apply the changes. This reduces the color depth and transparency values for the display.

3.3 Power Consumption

It might be necessary to make use of an *12 to 5 Volt converter* while connecting the Raspberry Pi with the boats power source. Since the power source on the boat may be limited, it makes sense to break down the power demands of the respective Raspberry Pi modules. Raspberry Pi requires a power supply with +5,1V. According to the official Raspberry Pi website[6], the amperage depends on the connected modules:

- **Raspberry Pi Model B**: minimum 500 mA
- **HDMI port** 50mA
- **Keyboard and mouse:** varies between 100mA or over 1000mA
- **GPS module**: about 60mA

Keyboard and mouse have the largest variance. Thus, the power demand specification of the keyboard and mouse are the main criterion for usage. In case the selected display has touch functionality, the mouse can be omitted. Combined keyboard and mouse modules with low power demand should be also considered.

The power supply can be bridged with a power bank that is capable of simultaneous charging and providing power. A sufficiently large power bank can supply the BoatPi for a whole sailing trip.

3.4 Supported Chart Formats

OpenCPN supports the following chart formats[7]:

- Worldwide standard S56 and encrypted S63 vector chart.
- BSB v3 and earlier raster charts.
- CM93 vector charts with per cell offset corrections
- ENCs, distributed by o-charts[8] for selected regions

[6] https://www.raspberrypi.org/documentation/hardware/raspberrypi/power/README.md
[7] https://opencpn.org/OpenCPN/info/about.html
[8] http://o-charts.org

When conversion from different formats is required, please refer to the Supplementary Software Manual[9] of OpenCPN.

4 Summary

In summary, we turned a Raspberry Pi and a GPS module into a chart plotter using OpenCPN. For more details on using OpenCPN, please refer to the OpenCPN User Manual[10]. The follow-up steps could include adding further components such as a weather monitor or a compass to the boat navigation system. In this case, the openplotter platform could prove helpful.

This guide is also available on GitHub[11]. It can be further extended by experiences from setting up and using the first prototype.

Christoffersen, Jens. 2016. "Make a Gps Navigation System for a Boat with a Raspberry Pi." www.allaboutcircuits.com.

Loschwitz, Martin. 2016. "Maritime Navigation with a Rasp Pi and Opencpn." www.raspberry-pi-geek.com.

[9] https://opencpn.org/wiki/dokuwiki/doku.php?id=opencpn:supplementary_software
[10] https://opencpn.org/wiki/dokuwiki/doku.php?id=opencpn:opencpn_user_manual
[11] https://github.com/lgalke/boatpi