Building a Smart Sailboat

Making a 30' sailboat a little smarter using a Raspberry Pi, a bunch of ESP8266 modules, and some Javascript.

Live version: rednightsky.com

The subject of our experiments in IoT:
1975 Yankee 30 MKIII.

Hull Type:	Fin with rudder on skeg
Rig Type:	Masthead Sloop
LOA:	30.04' / 9.16m
LWL:	25.00' / 7.62m
Beam:	9.00' / 2.74m
Displacement:	10000 lbs./ 4536 kgs.
Ballast:	4850 lbs. / 2200 kgs.
Designer:	Sparkman & Stephens

The Components

Sensors, Sensors, and more Sensors

- Anyone who has ever worked on a small sailboat knows the hell-scape of pain that is contorting ones self into wild positions inside every dark nook and cranny to run wire or drill holes or what have you. When I initially started to prototype out a network of sensors to monitor different systems all over the boat, it quickly became clear wireless was the way to go. So I began with the venerable ESP8266 micro-controller and started to design a simple system around it. The ESP8266 has a full wifi network stack, plenty of IO pins, and can be programmed with the Arduino IDE. Simple, easy. I got mine from Adafruit. Depending on what I'm monitoring, they are paired up with a voltage & current sensor, an environmental sensor, or just reading voltages from one of the analog pins. Each sensor and ESP is soldered on to a small circuit board and housed in a weatherproof enclosure.

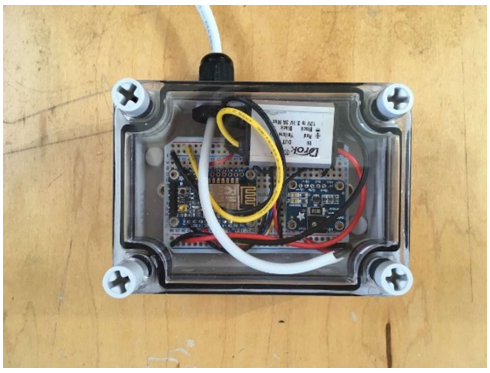
The wireless sensors monitor:

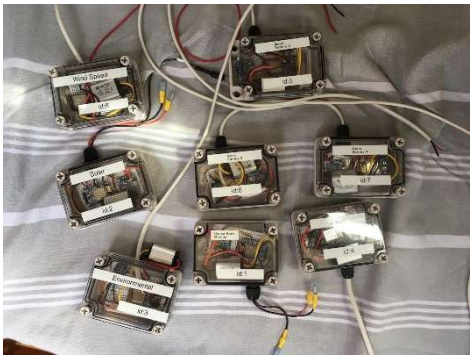
- House battery bank, which supplies 12 volt DC power to the lights, radio, gps, and refrigerator.
- 4 separate motor batteries. I installed an electric motor in place of the old gasoline engine. The batteries are wired in series to create 48 volts DC.
- Real-time energy usage.
- Solar energy creation. A Blue Sky Energy Solar Boost 3000 MPPT controller handles power input from 3 removable solar panels.

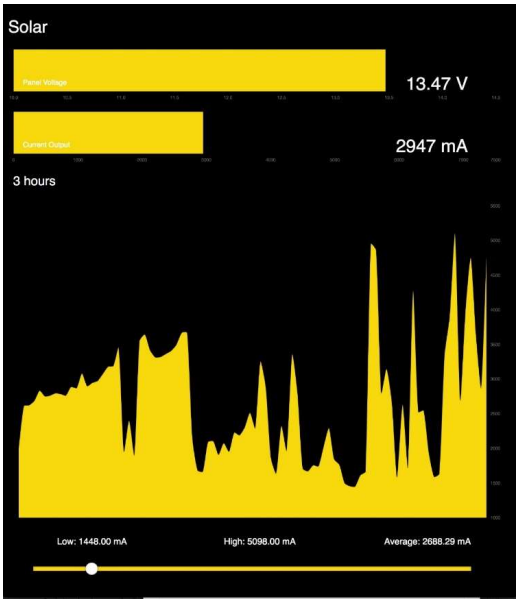
- Environmental conditions. Temperature, atmospheric pressure, and humidity are gathered from a Bosch BME280 breakout board. Wind speed is calculated from an analog voltage produced by a anemometer mounted at the mast head. For this sensor/module I used the ESP32, both to try it out, and because it has a higher resolution ADC. 12bit as opposed to the ESP8266's 10bit.

Motor and GPS data is brought in over USB.This bounty of data riches is finally all ingested by a Raspberry Pi running software I wrote in Node.js. gives me all the data. (Wind speed sensor was later folded into the Environmental sensor)

Solar power info displayed in the app.







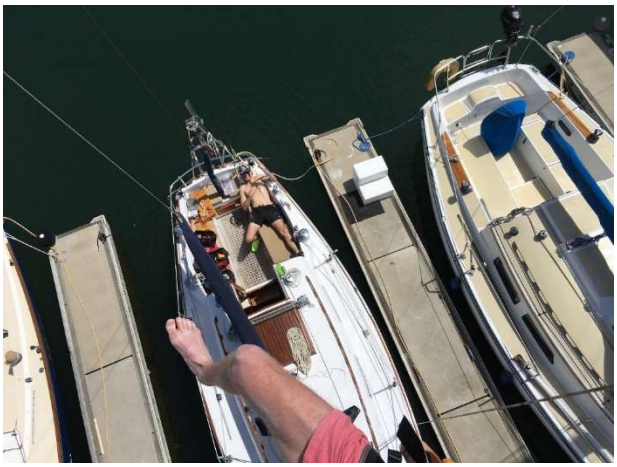
The Raspberry Hub Node.js

The on-board Raspberry Pi has a number of jobs to do and a server written in Node is the task master. It listens for any sensors in wifi range and if discovered opens up a TCP connection and starts getting data. As a system, a couple of my major design goals were to implement a database and to use React as the front-end framework. I realized early on if each sensor outputted a nicely formatted JSON packet it would make everything down the line easier. Once the Pi receives sensor data it has to parse the JSON, merge it with other sensor data, insert it into the database, serve up a local web application, and retransmit the data out to an internet accessible machine for remote viewing. I use Systemd to start and keep the server process running.

MongoDB

For the database, I run MongoDB on the Pi and, so far, it has worked very well. The Pi uses an SD card to store the disk image. Because SD cards weren't designed for the constant read/writes of something like a database, I use a 16gb USB thumb drive as the Mongo data store. Currently I'm only logging the last 24 hours of data, so I setup a Time To Live index on the Mongo collection which will expire data after a day, keeping the drive from filling up and making the db snappy.

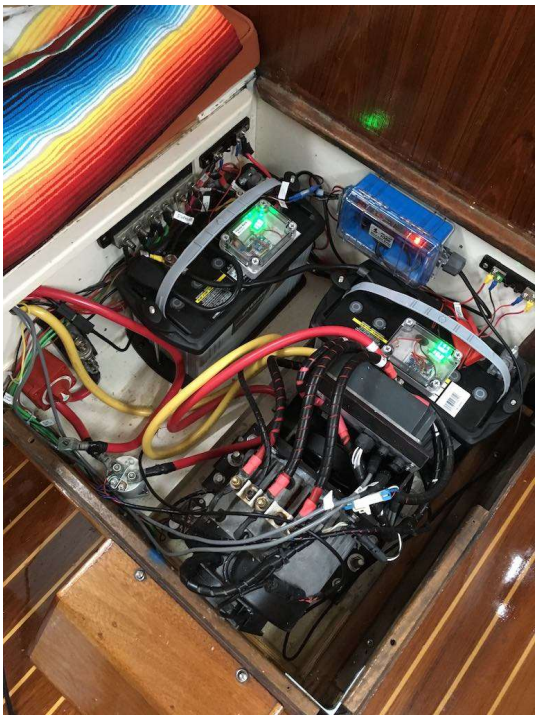
Networking



They say it's important to network. The Pi creates a local net using it's on-board wifi and serves up the main web application to any local device. It also sends data out to an internet reachable machine. I mounted a Ubiquiti Bullet M2 wifi radio with a high gain antennae at the top of the mast. I was able to feed 50' of cat6 cable down the mast to the Pi. If connected to a wifi network, the Pi will transmit sensor data out over websockets. Using a DHCP server and some iptables configuring, the Pi acts as a router for the Ubiquiti modem and bridges internet access, creating a nice strong wifi network for the boat. For those times when i'm not in a marina surrounded by wifi, I installed a Huawei USB cell modem and use a hologram SIM card to transmit data via 3G cellular. Getting high mounting the WiFi radio on the mast.

The Raspberry Pi is housed in the blue Pelican Case. The two 100 amp/hour batteries beside the motor are the house bank.

Electric Motor



After it caused a number of strandings, leaks, fires, and near sinkings, I finally removed the original Atomic 4 gasoline engine and replaced it with an electric motor. Electric Yacht sent me a 10kw 48 volt motor kit that I promptly set about trying to hack in to. To run the thing, a 48 volt battery bank is created by wiring four 235 amp/hour batteries in series.

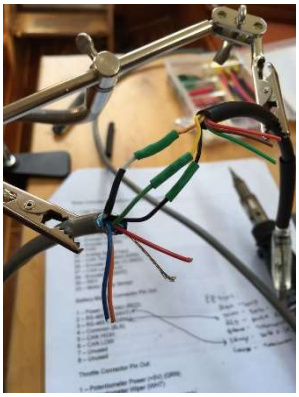
3 of the 4 motor batteries before being installed. They are heavy.

Each battery has it's own separate sensor which helps keep track of the balance of voltages. If they get too

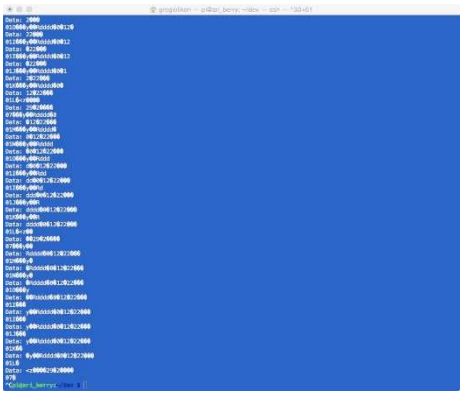
far out of whack with each other charging capacity and life expectancy can suffer Starboard side motor batteries installed.

The system came with a wired display that mounts in the cockpit and plugs into a controller on the motor. While wiring it up, It's hard to miss the second, unused communications port. Is there nothing that piques a nerd's interest more than an open comm port? I think not! So I set about trying to extract some data from it. It proved to be a lot easier than I thought.

The physical electrical interface is RS-485. I was able to hack together a home made cable from the plug end of an extra broken motor monitor I had and a USB to RS-485 cable I found on A mazon. Using the Serialport module for Node, getting data was damn easy. I love me a good heat shrunk connection.

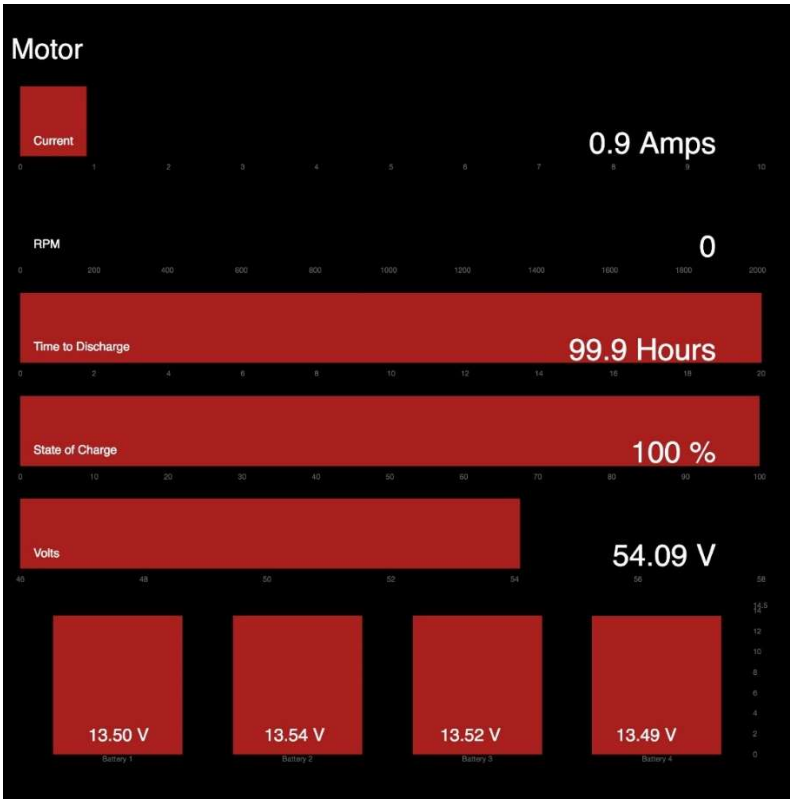


I emailed Electric Yacht and told them what I was up to and they sent me a byte level breakdown of their proprietary protocol. Amazingly nice of those guys.

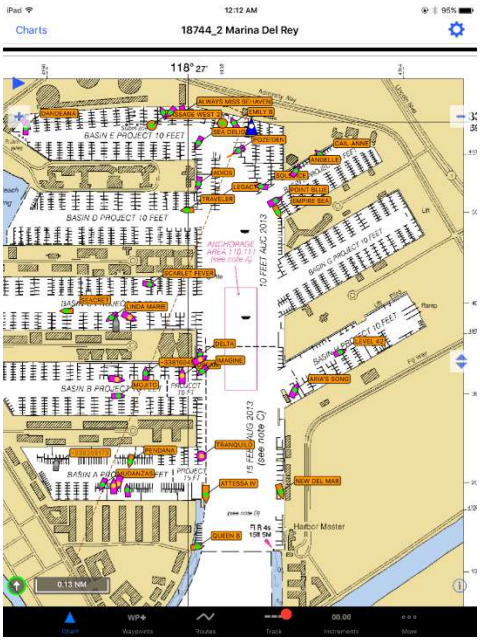


At first, garbage. But garbage with structure!

Once we look at it in hexadecimal and get our port speed dialed in, it's clear we have something: After that, it was just a matter of some simple bit shifting here and there in Node to get our values right. Now we have access to our electrical motor data and can integrate it into our sensor telemetry and view it all wirelessly on any web capable device.



GPS



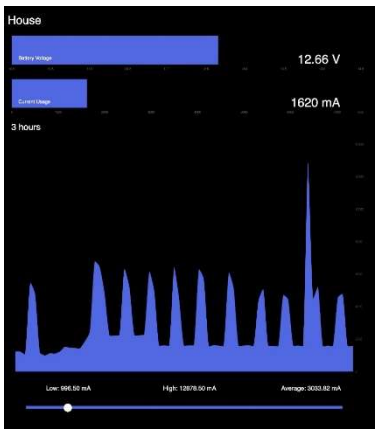
A GPS antennae is wired into an AIS receiver which multiplexes the data streams and transmits it as NMEA 0183 messages over USB. Using Kplex running on the Pi, it is brought into Node and also retransmitted on the local wifi network so we can use chart plotting software on the iPad, or any web connected device that can make use of the data.

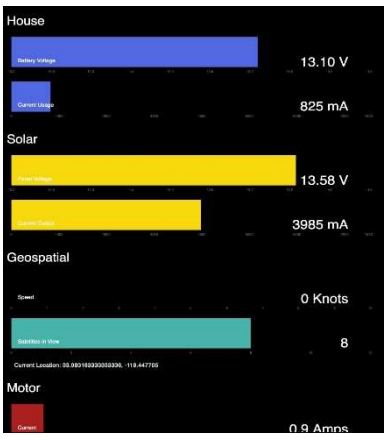
AIS targets and position on an iPad from multiplexed NMEA data over wifi: Raw messages coming in:

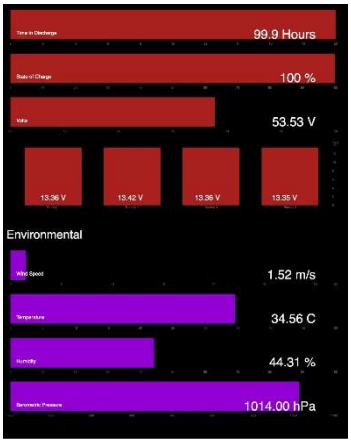
Webpage

The last piece was creating an internet accessible VPS server running a web application built with Node.js, Express, MongoDB, and React. It receives info from the boat as JSON data over websockets and renders real-time data quickly. I built a simple API that pulls historical data from the Mongo database and graphs it over time.

Here is the last 3 hours of energy usage. The spikes are the refrigerator compressor kicking on.







The Result

A fast, single page app that gives us real-time status and historical info of 20 data points on a sailboat.