Raspberry Pi Geocaching System (Pi Hiker)
Author: Bob Alexander

A kid friendly, high accuracy geocaching platform.

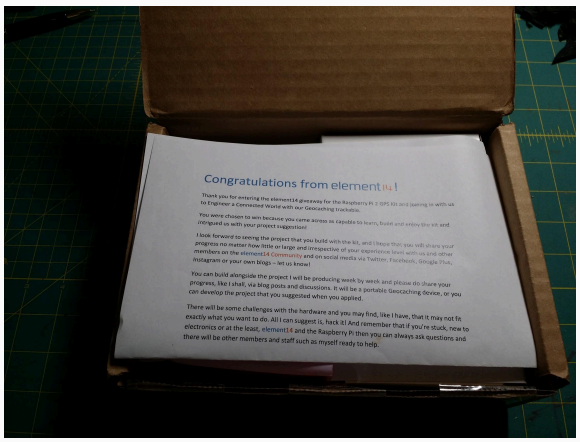
- Accuracy of the GPS location could be improved by putting the L80 into 10Hz mode and then performing an averaging function such as outlined in this paper:

  An Effective Approach to Improving Low-Cost GPS Positioning Accuracy in Real-Time Navigation
- With the addition of a magnetometer, a direction indicator could be added that would not be dependent upon movement to show the target azimuth.
- We would utilize our Makerspace's 3D printer to print an enclosure for this platform to sit in such that it could be used by children.
- I think the Pi 2 is a good candidate for this project as it should have the horsepower to perform the averaging function and be able to utilize a database of geocaches downloaded before the trip.

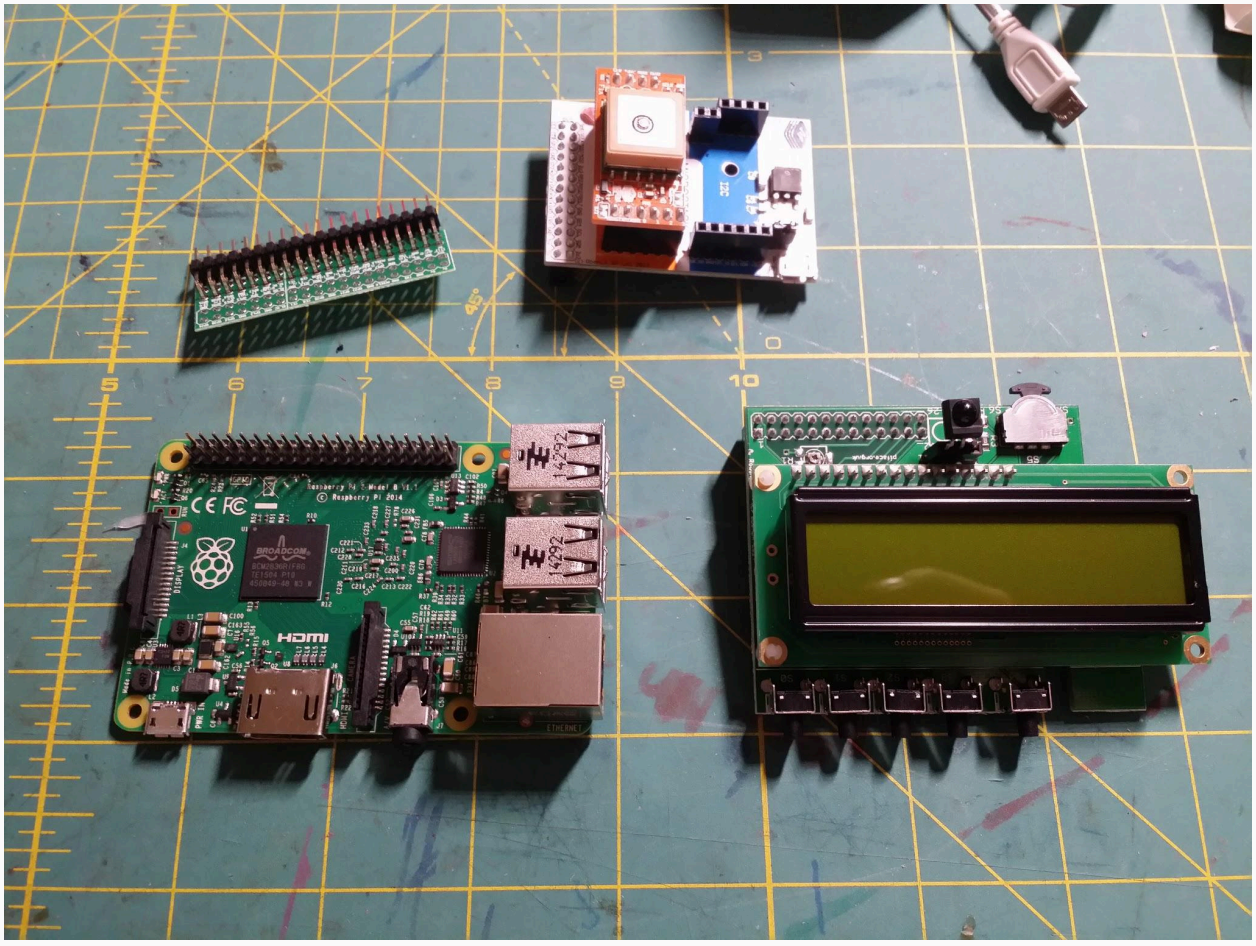
Journal:

Received notification that I won a Raspberry Pi 2 GPS kit from Christopher Stanton on July 17th 2015.
Received kit in mail August 4th 2015.

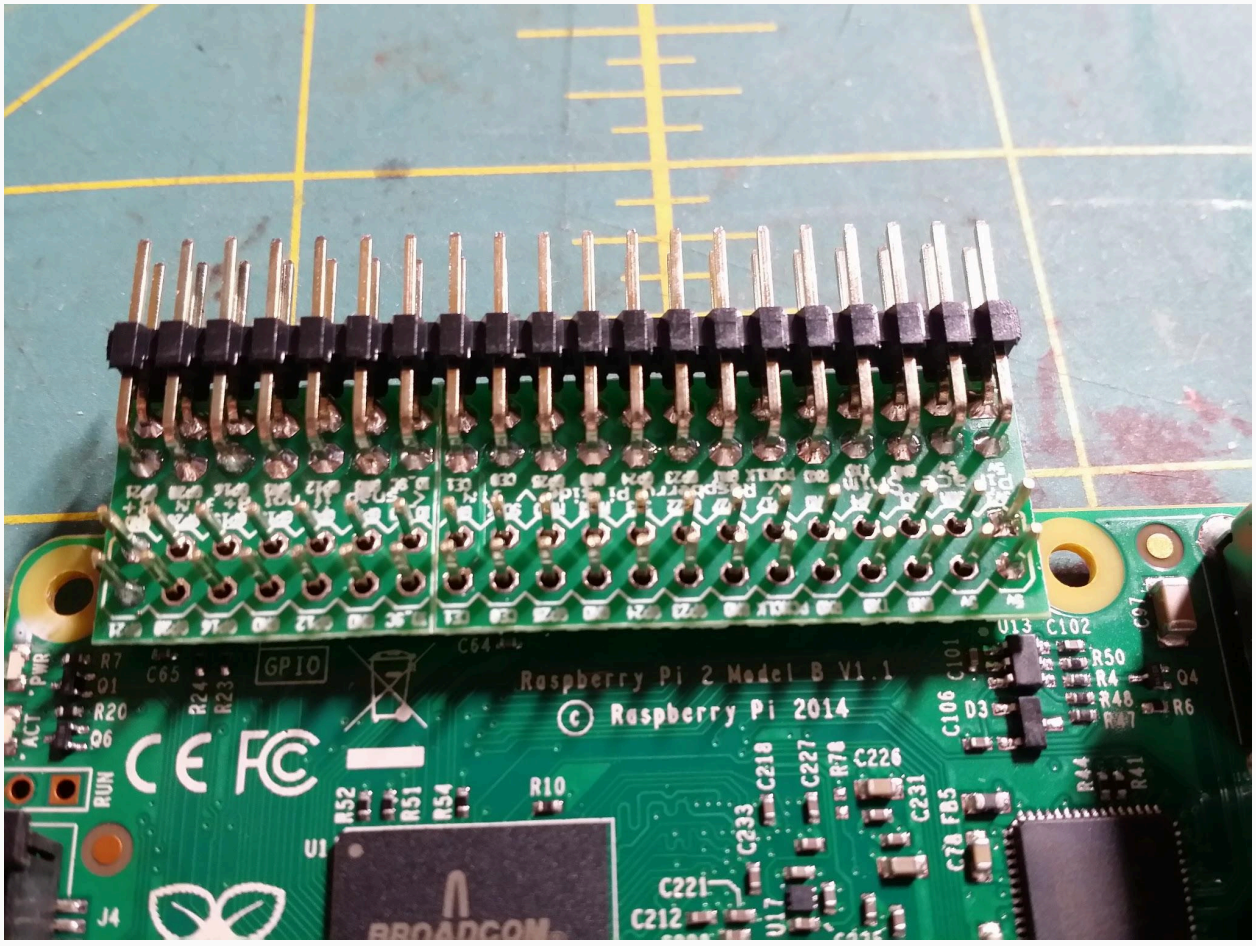
:)



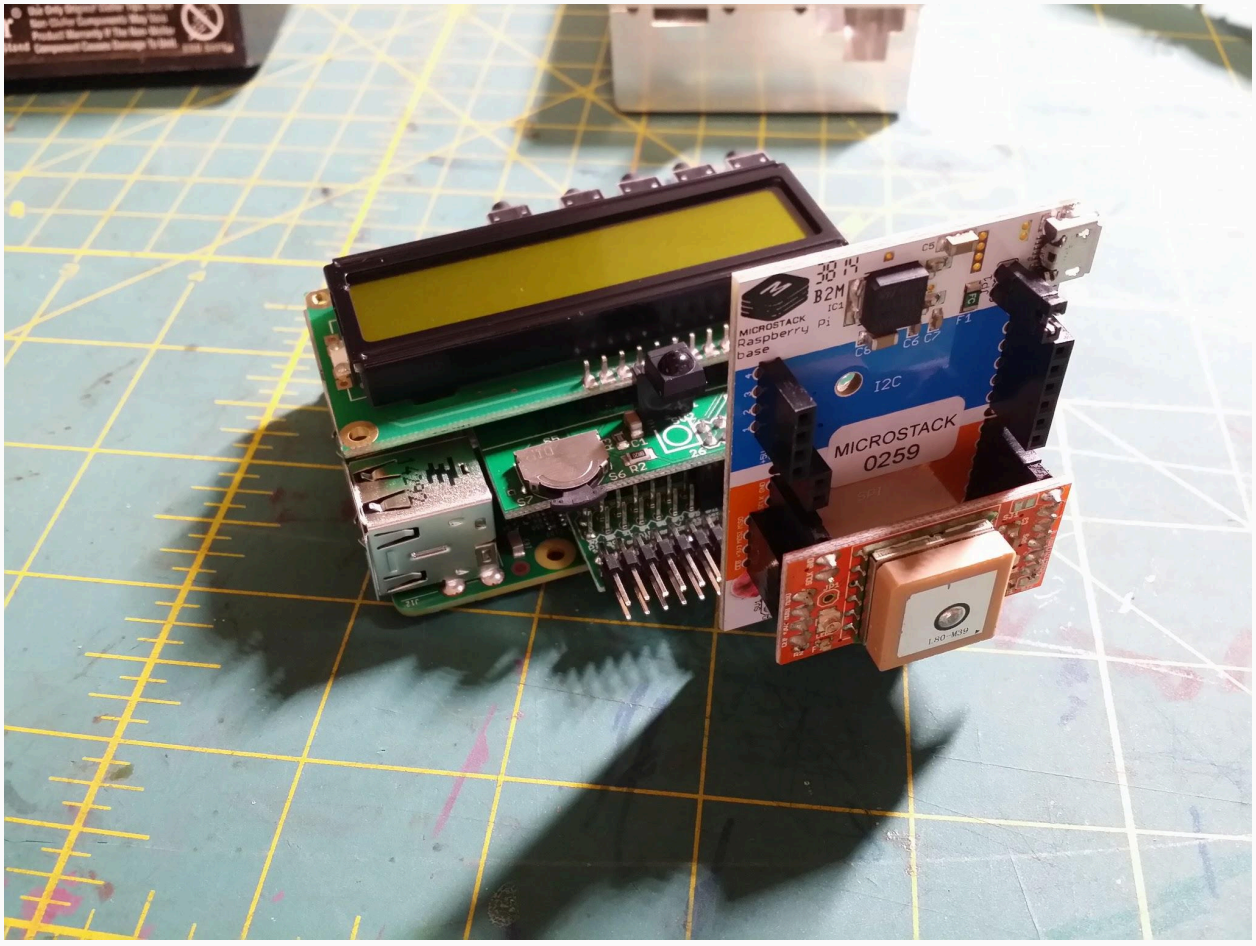
Look at all the goodies!!



Starting to assemble...



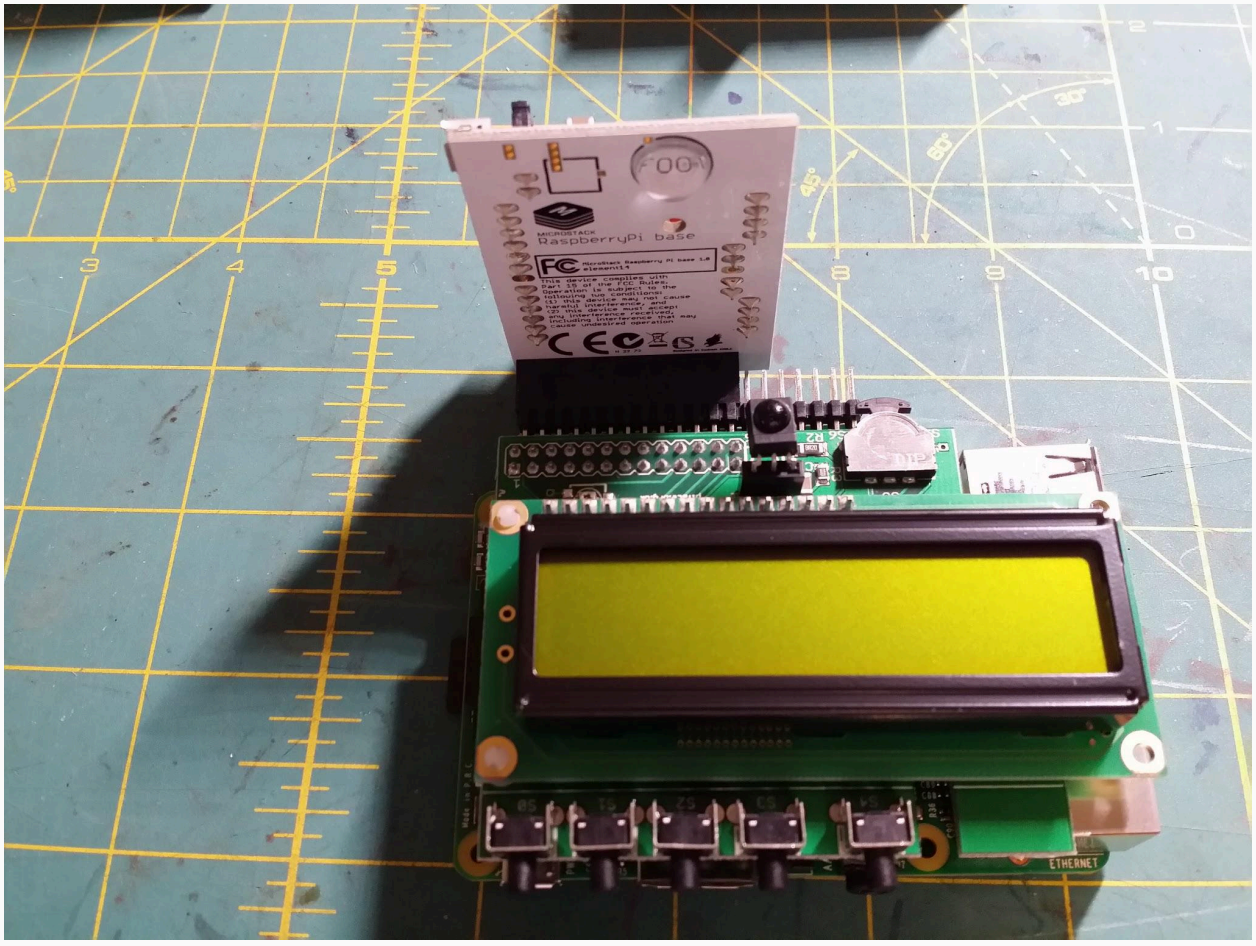
I soldered the right angle header pins to the PiFace GPIO Duplication Board Shim and tacked the shim in place on the Raspberry Pi 40 pin header board by soldering the four outermost pads as I applied pressure against the shim. It was a little loose and didn't maintain good contact without this. Yes, I soldered the pins on the wrong side and the silk screen is backwards. Oh well. A quick assembly sheet would be nice to prevent this as it wasn't obvious to me - NOOB!



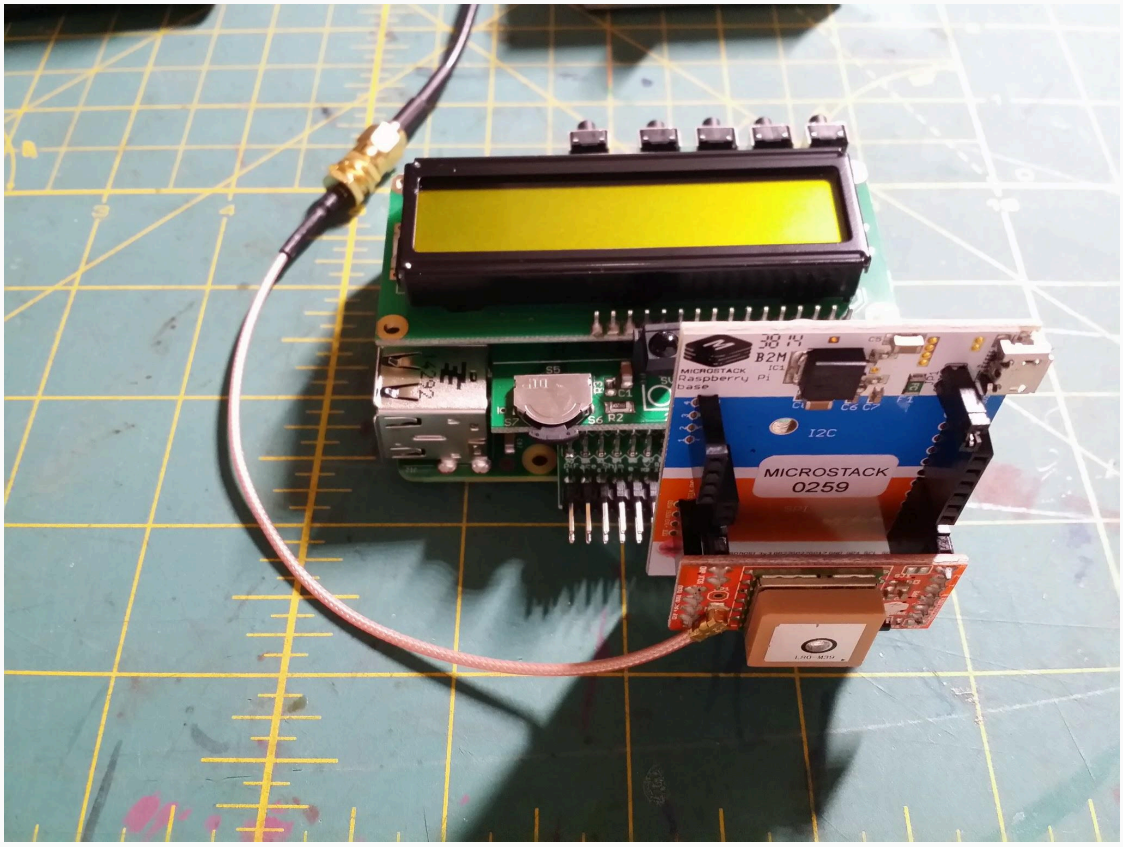
The footprint is a lot bigger than I expected! How am I going to house all this?!

The Microstack is definitely a nice system for prototyping but I don’t think it will work for a ‘finished’ project.

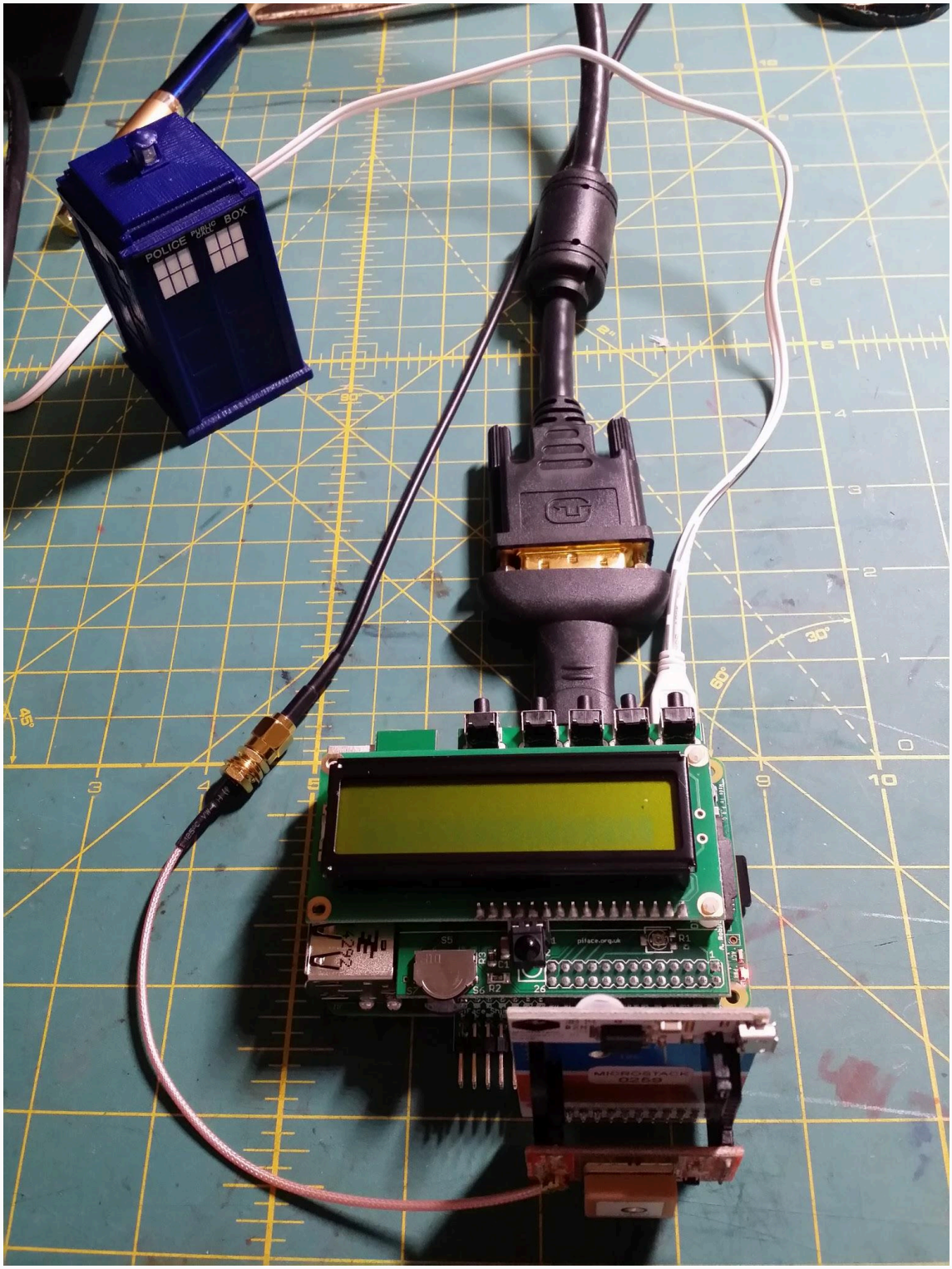
Ben Heck style chopping/hacking in store?



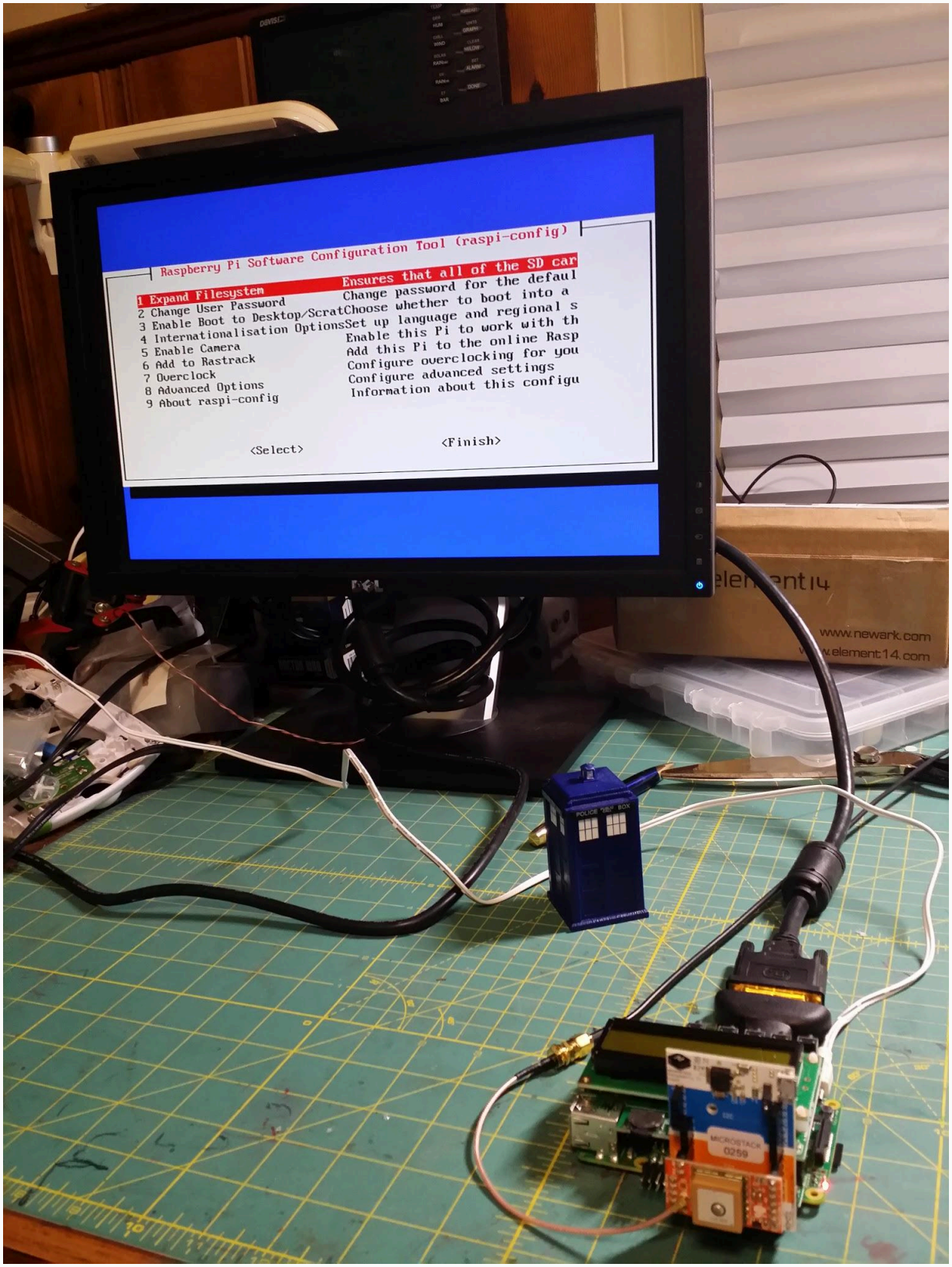
Maybe a game pad style case will accommodate the assembly as is?



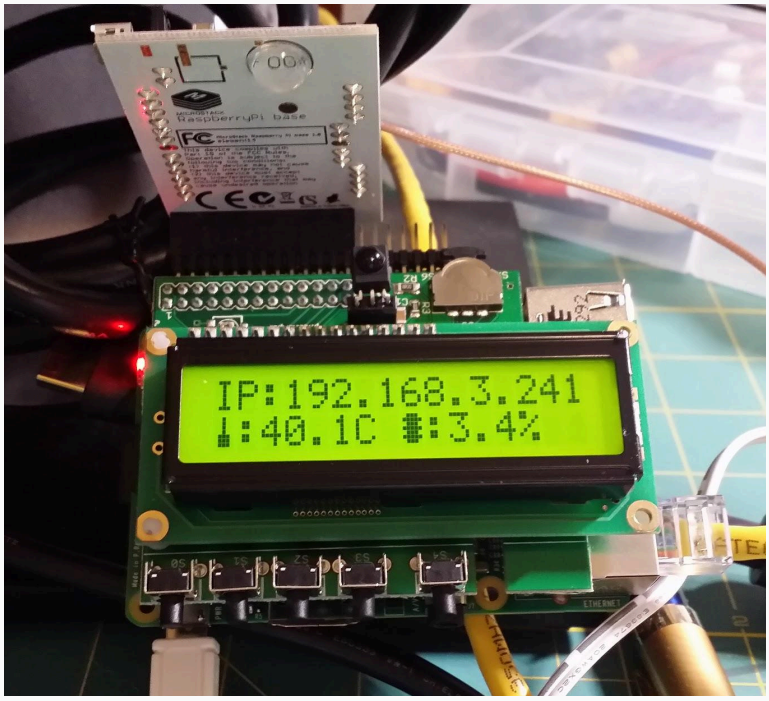
Connected to GPS antenna sitting on the window sill. Retractable reel for the GPS antenna?



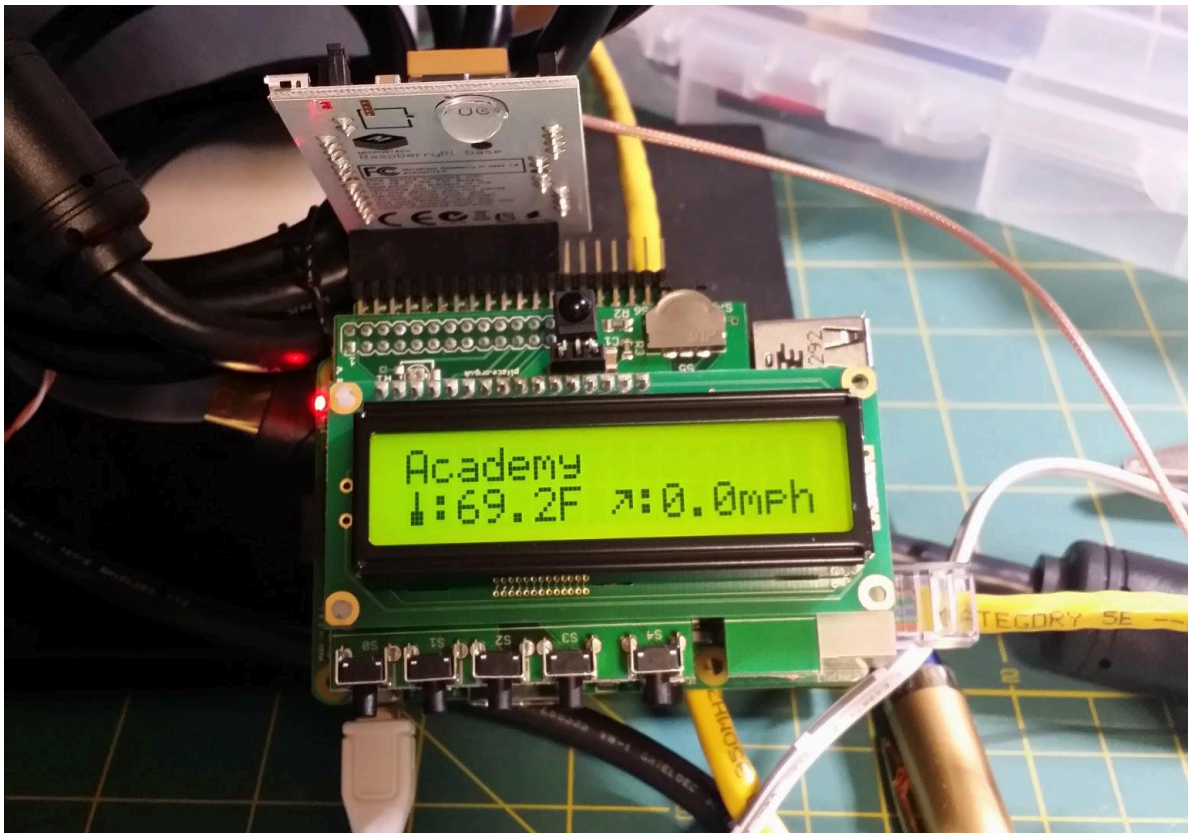
Ready to power up! Note the DVI-HDMI adaptor for my DVI only monitor.


Raspberry Pi Software Configuration Tool (raspi-config)
1 Expand Filesystem
2 Change User Password
3 Enable Boot to Desktop/Scrat
4 Internationalisation Options
5 Enable Camera
6 Add to Rastrack
7 Overclock
8 Advanced Options
9 About raspi-config
Ensures that all of the SD car
Change password for the defaul
Choose whether to boot into a
Set up language and regional s
Enable this Pi to work with th
Add this Pi to the online Rasp
Configure overclocking for you
Configure advanced settings
Information about this configu
<Select>
<Finish>
www.newark.com
www.element14.com

It works!



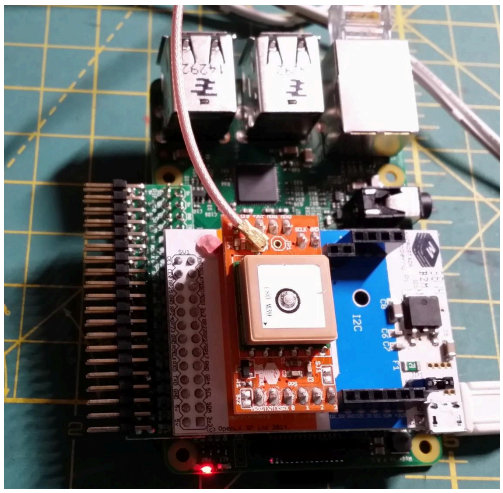
After downloading the [PiFace Control and Display] software and enabling the SPI bus I ran the sysinfo demo.



Just a quick tweak and it displays my local weather!

August 5th 2015: GPS testing
Following along with Christopher Stanton's blog:
Putting Together the Pieces - Raspberry Pi 2 GPS Geocaching Project

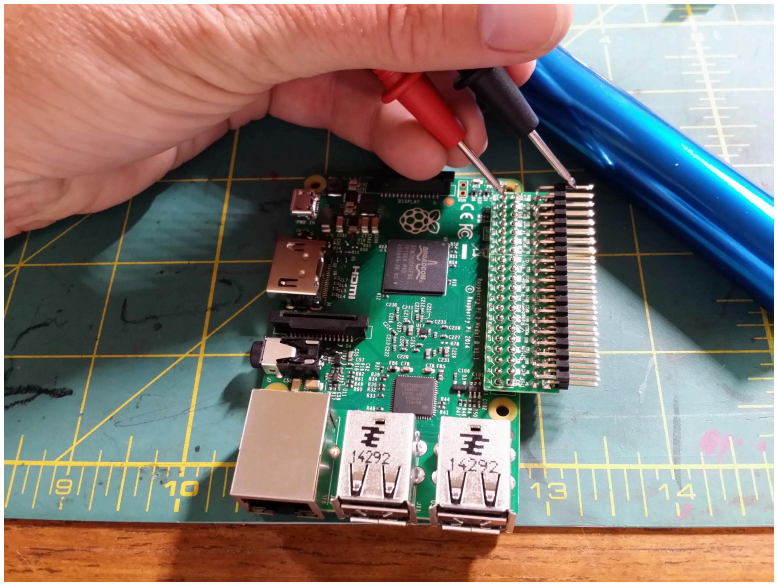
After not having luck with
cat /dev/ttyAMA0
and 'cgps -s' just returning "cgps: GPS timeout"
I unplugged the PiFace CAD and moved the GPS Microstack to the direct connection instead of going through the shim (after powering off the pi).



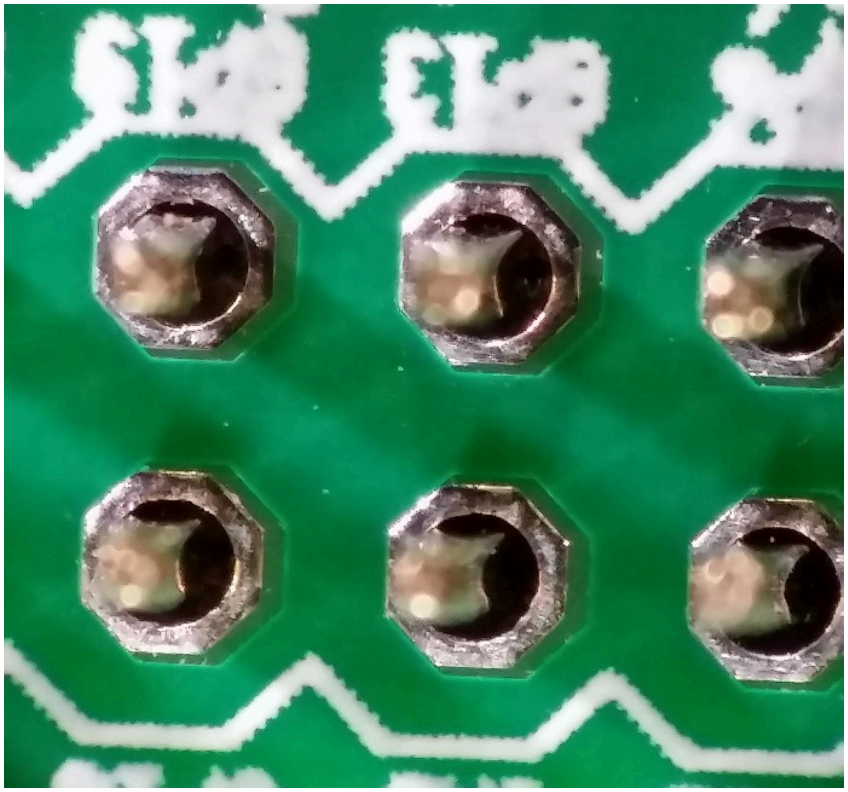
After booting back up and waiting for the GPS to initialize, I reran 'cgps -s' and BINGO!

```
┌──────────────────────────────────────────┐ ┌──────────────────────────────────┐
│    Time:       2015-08-06T02:07:10.000Z  │ │PRN:   Elev:  Azim:  SNR:  Used:  │
│    Latitude:    3X.650558 N              │ │  15    78    169    41      Y    │
│    Longitude:   7Y.667636 W              │ │  20    71    001    32      Y    │
│    Altitude:   214.9 m                   │ │  21    45    311    29      Y    │
│    Speed:      0.4 kph                   │ │  29    44    220    36      Y    │
│    Heading:    252.9 deg (true)          │ │  18    26    274    36      Y    │
│    Climb:      0.0 m/min                 │ │  24    08    163    00      N    │
│    Status:     3D FIX (1 secs)           │ │  43    00    000    00      N    │
│    Longitude Err:   +/- 173 m            │ │                                  │
│    Latitude Err:    +/- 20 m             │ │                                  │
│    Altitude Err:    +/- 478 m            │ │                                  │
│    Course Err:      n/a                  │ │                                  │
│    Speed Err:       +/- 1251 kph         │ │                                  │
│    Time offset:     -0.511               │ │                                  │
│    Grid Square:     FM08pp               │ │                                  │
└──────────────────────────────────────────┘ └──────────────────────────────────┘
```

Now, I just need to troubleshoot the shim…
Even though I soldered the connector on backwards shouldn't the one for one pinout be the same? Or, maybe the serial port connections are not making contact?
Time to ohm it out and check!

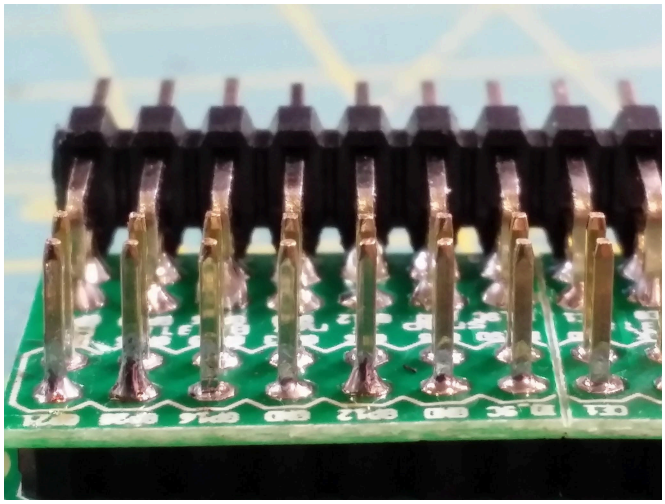
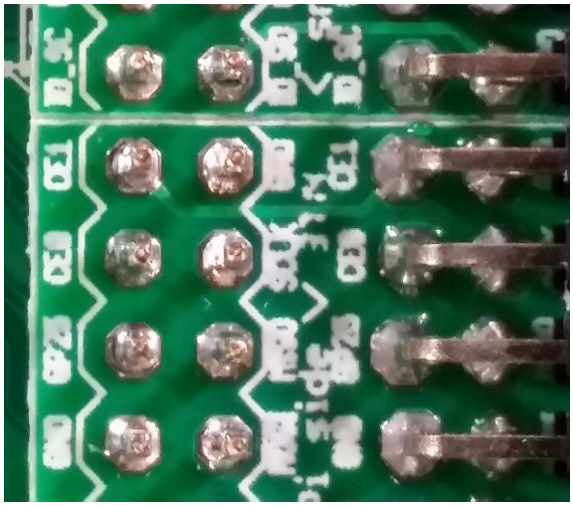


Nope, pinout is fine - but the gap around the solder pads on the shim and the pins themselves seems to be too big and is not making a connection with the board.



This is probably a symptom of me soldering the right angle pins on the **wrong side** of the shim. Perhaps the holes drilled and plated on the other side of the shim (where the right angle pins are now) are smaller? Rather than desolder the shim/header pins and do it all over again (never have had much luck desoldering right angle pins without destroying the board and the pins) I just soldered the shim VERY carefully to the Pi's headers with a tiny amount of solder so as to

not interfere with the rest of the GPIO pin on the board itself. I want it to be reliable and at least somewhat durable anyways! A quick run through with the continuity tester verified this fixed the problem.

Reassemble and power back up the system and continue with the testing!

The cgps command checks out - I get a fix! :)
Running the weather python script displays the data to the screen but it isn't acting the same.
The buttons/city toggle aren't working now! :(
Unplugging the microstack/GPS module doesn't seem to help matters any.
Looks like more troubleshooting is needed. Perhaps the shim/my soldering is preventing the CAD from seating all the way? It looks fully seated.

Running through the print(cad.switches[x].value) in Python while toggling the switches seems to be working fine. I'm not sure what has changed at this point as all of the buttons seem to be working but the weather.py demo program is not. Moving on!

August 6th 2015: GPS output to LCD screen
Referencing:
Microstack GPS and Piface Control and Display Geocaching by callum smith
This code will simply get the coordinates and time from the GPS and display them on the LCD:

```
#!/usr/bin/env python3
"""
Grab data from GPS and spit it out to LCD
"""

import sys
PY3 = sys.version_info[0] >= 3
if not PY3:
    print("Please use `python3`.")
```

```
    sys.exit(1)

###import libraries and set up gps
from time import sleep
import pifacecommon
import pifacecad
import sys,math,time,microstacknode.gps.l80gps
gps=microstacknode.gps.l80gps.L80GPS()
cad=pifacecad.PiFaceCAD()
cad.lcd.backlight_on()
cad.lcd.cursor_off()
cad.lcd.blink_off()
cad.lcd
cad.lcd.home()
cad.lcd.write("Obtaining pos...")

while True:
    cad.lcd.home()
    current_pos=False
    while current_pos==False:
      try:          # try command used to prevent crash when no
response from GPS
          current_pos=gps.gpgll          # gets current GPS position
      except (microstacknode.gps.l80gps.DataInvalidError,
microstacknode.gps.l80gps.NMEAPacketNotFoundError):          # if no
GPS response or if response is invalid
          time.sleep(1)                    #pauses before retry to
connect to GPS
          cad.lcd.write("No GPS Lock\nTry moving or check ant")

cad.lcd.write(str(current_pos["latitude"])[:7]+","+str(current_pos["l
ongitude"])[:8]+"\n"+str(current_pos["utc"]))
    time.sleep(1)                    #pauses before looping to allow
user to read output
    #cad.lcd.clear()  #will cause flickering but will ensure no mess
is left on screen
```

August 12th 2015:
This code will grab the coordinates from the GPS and stick them into an array. Once 10 values have been accumulated it will spit out the average and standard deviation.
This is a VERY basic averaging and not the permanent or intended method of obtaining a 'high accuracy' reading. In this mode it takes 10 seconds for the averaging, I intend to speed up the GPS output to 10Hz which will give a simple average once per second.
The goal of this method is to provide a baseline (which will be logged to a CSV file by the Pi) to compare methods of obtaining a highly accurate location.
This method will be called 'Simple Averaging' and will be compared with 'Recursive Averaging', the Kalman filter, the Wiener filter, autoregressive-moving average (ARMA), and the method outlined in the paper referenced above which improves position while moving. Once the data from these methods is graphed I will decide which method to implement for the geocaching tool.

```
#!/usr/bin/env python3
"""
Grab data from GPS and average it over 10 data points
"""

import sys
PY3 = sys.version_info[0] >= 3
if not PY3:
    print("Please use `python3`.")
    sys.exit(1)

###import libraries and set up gps
from time import sleep
import numpy
import sys,math,time,microstacknode.gps.l80gps
gps=microstacknode.gps.l80gps.L80GPS()
lat_pos = numpy.zeros(10) #initialize latitude array
lon_pos = numpy.zeros(10) #initialize longitude array

current_pos=False
while current_pos==False:
    try:          # try command used to prevent crash when no
response from GPS
        current_pos=gps.gpgll          # gets current GPS position
```

```
    except (microstacknode.gps.l80gps.DataInvalidError,
microstacknode.gps.l80gps.NMEAPacketNotFoundError):          # if no
GPS response or if response is invalid
            time.sleep(1)                    #pauses before retry to
connect to GPS
count=0
lat_pos.itemset((count),current_pos["latitude"])
lon_pos.itemset((count),current_pos["longitude"])
while True:
    if count == 9:
        print(lat_pos)#+" STD: "+numpy.std(lat_pos)
# for debugging
        print(lon_pos)#+" STD: "+numpy.std(lon_pos)
# for debugging
        print(numpy.mean(lat_pos,
dtype=numpy.float64),numpy.std(lat_pos))
# for debugging
        print(numpy.mean(lon_pos,
dtype=numpy.float64),numpy.std(lon_pos))
# for debugging
        count=0
    else:
         count += 1
    try:          # try command used to prevent crash when no
response from GPS
        current_pos=gps.gpgll          # gets current GPS position
    except (microstacknode.gps.l80gps.DataInvalidError,
microstacknode.gps.l80gps.NMEAPacketNotFoundError):          # if no
GPS response or if response is invalid
            time.sleep(1)                    #pauses before retry to
connect to GPS
    lat_pos[count]=current_pos["latitude"]
    lon_pos[count]=current_pos["longitude"]
```

Published with edits to:
http://www.element14.com/community/community/raspberry-pi/raspberrypi_projects/geocaching/blog/2015/08/13/pi-hiker--raspberry-pi-geocaching-system

https://github.com/adafruit/Adafruit-GPS-Library/blob/master/Adafruit_GPS.h
http://www.farnell.com/datasheets/1860443.pdf
http://www.adafruit.com/datasheets/PMTK_A11.pdf

To set 5Hz mode:

First change the Baud rate

To be able to go above 1Hz you will need to switch to a higher baud rate:

```
echo -e "\$PMTK251,57600*2C\r\n" > /dev/ttyAMA0
```

Don't forget to update your tools connection speed to 57600!

Then

```
echo -e "\$PMTK220,200*2C\r\n" > /dev/ttyAMA0
```

for 5Hz mode

To go back to 9600bps:

```
echo -e "\$PMTK251,9600*17\r\n" > /dev/ttyAMA0
```

Supported speeds:

4800,9600,14400,19200,38400,57600,115200

Other update rates:

100mHz "$PMTK220,10000*2F" // Once every 10 seconds, 100 millihertz
200mHz "$PMTK220,5000*1B" // Once every 5 seconds, 200 millihertz
1HZ "$PMTK220,1000*1F"
5HZ "$PMTK220,200*2C"
10HZ "$PMTK220,100*2F"