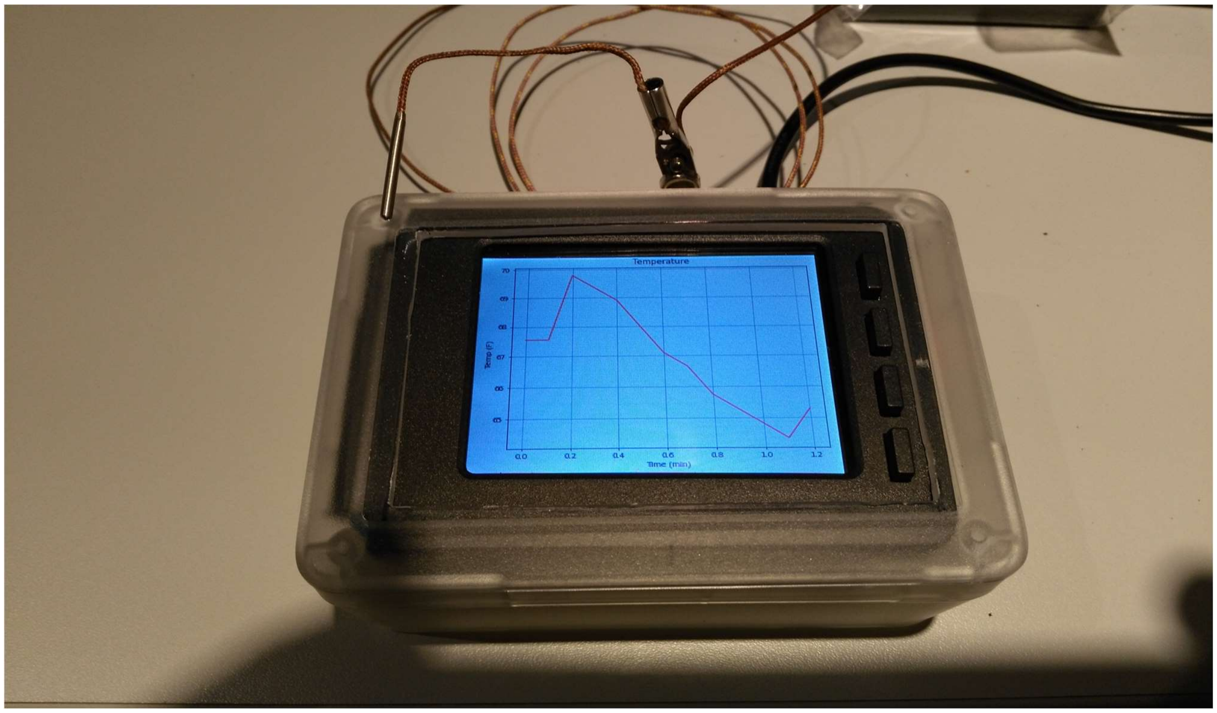
World's Greatest Oven Thermometer

(Rev 0.7: June 11, 2019)



Apologies for the moire effect in this photo

Features

On the the stacked 2.8" PiTFT capacitive touchscreen, this device will show the current temperature, with the option of an additional save/hold temperature value or even a graph of the temperature over time, using a cable-attached type K thermocouple probe.

The advantage of using a thermocouple is that it can measure very high and very low temperatures with good accuracy – and the Pi assembly itself doesn't have to be exposed to them.

Optionally, this Linux-based device can be shutdown and/or restarted at the touch of a single button, to avoid having to remove and replace the power supply connection.

There is also a handy flashing LED activity indicator, so that you can check if the software is alive: especially useful in "headless" mode.

No Internet connection is required, because all of the timing is relative.

It even creates a csv log file that can later be used offline, to build fancy graphs or spreadsheets for analysis purposes and typical CPU usage is below 10%. The last two csv log files are saved, so that data from a recent test isn't lost after a reboot.

For development purposes, the software can be tested on an attached X11 HDMI screen/keyboard (or via VNC) without access to the PiTFT itself, by un-commenting a single Python statement, though you definitely need the Perma-Proto Hat board with the MAX31855 board attached.

Background

Why make an oven thermometer using a quad core, general purpose computer with a multi-user operating system? Because you can! And because of all the pieces together cost about a hundred bucks, and because all of the open source software is totally free.

When we got a new kitchen range, it didn't bake as well as the old one, so I tried a $7.99 oven thermometer – no help, since it was so very slow to respond. Then I tried a multimeter with a temperature probe, but I had to write down all of the values, over time, and graph them to a spreadsheet – too boring, time consuming and error-prone.

It was only when I cobbled together a Raspberry Pi and an Adafruit MAX31855 breakout board with a thermocouple probe, along with some copy/paste Python programming and, Matplotlib, that I uncovered the real problem. The oven's temperature was dipping down significantly from the set temperature for 10 minutes after the oven said that it was pre-heated – not good for cookies that only take 20 minutes to bake. So now we use a 20 minute pre-heat, regardless of what the oven says.

Problem solved? Well yes, except that we had friends with similar issues and it was a pain to drag my screen, keyboard, breadboard, Pi and cables over to their home and set it up.

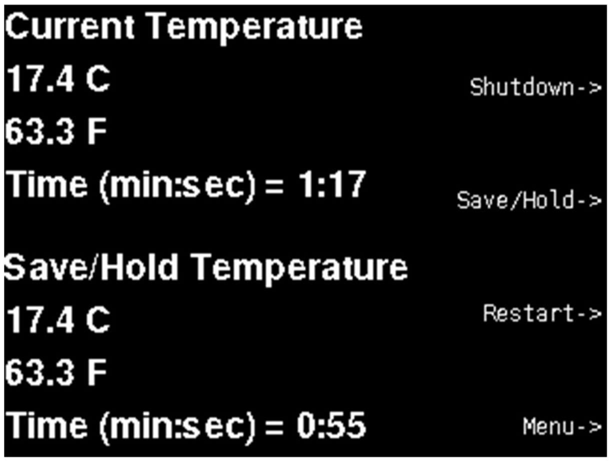
So, in the interest of portability I built a prototype, self-contained unit that can be easily used to measure and graph oven temperatures, using a Raspberry Pi 3 B+ with Raspbian Stretch, an Adafruit MAX31855 thermocouple breakout board, an Adafruit 2.8" capacitive touschscreen, along with Python, Pygame and Matplotlib (amongst other open source packages.) All you'd need to carry around is the main unit and a power supply.

The following describes how it works and how it's made.

Please keep in mind that this is just a prototype, eh.

Operation

When first powered on, the device will display a "Starting" splash screen, followed by a continuously updated time/temperature display with button labels.



```
Current Temperature
17.4 C                         Shutdown->
63.3 F
Time (min:sec) = 1:17          Save/Hold->
Save/Hold Temperature
17.4 C                         Restart->
63.3 F
Time (min:sec) = 0:55          Menu->
```

"Buttons" referred to below are referenced as button 1 being labelled #17 on the PiTFT board, button, 2 as #22, button 3 as #23 and button 4 as #27".

Button 1 is a software-independent OS shutdown/restart button, which will perform an orderly shutdown of the OS, when the device is active, or will reboot it if it has been shutdown. The latter saves having to disconnect and reconnect power to the Raspberry Pi to restart it after a shutdown.

The following are screenshots of the currently available display options, captured in X11 testing mode using the built-in Raspbian Stretch "scrot" command. Type "man scrot" in a terminal window to see how "scrot" works.

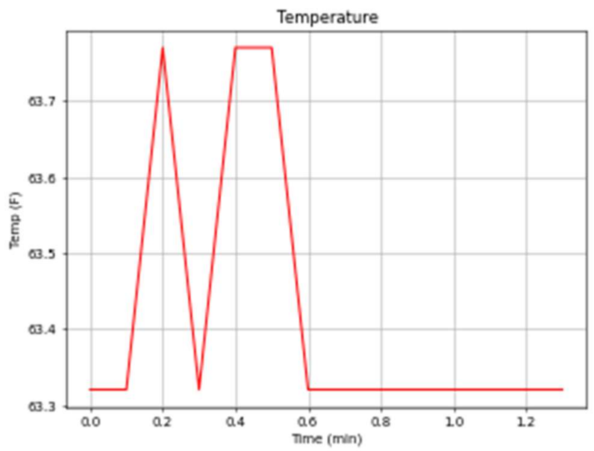
At any time, pressing "x" on the keyboard, or clicking "X" on the X11 window title bar will just exit the software – the current prototype debugging default.

Also, in the above display, pressing button 2 on thePiTFT (or the "h" key on the keyboard) will capture the current temperature and time as a Save/Hold Temperature and update the display.

Similarly, pressing button 3 (or "r" on the keyboard) will restart the timer and temperature capture data. A line of zero ("00") values will be written to the csv file to record this event for future analysis purposes.

Graph Display

Touching the PiTFT screen (or clicking directly on the X11 window) during the above display will switch to the following graphic display.



Axis values are automatically adjusted based on the total time period and temperature range that the graph covers, so decimal points may move or disappear and the graph line colour will change to blue if the temperature is at or below freezing.

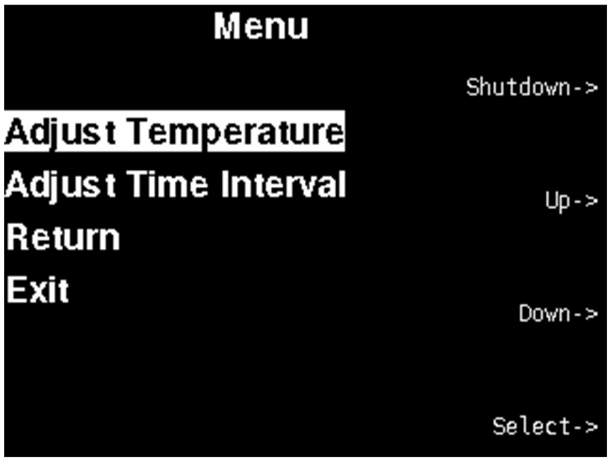
Touching the PiTFT screen (or clicking on the window in X11 mode) while the graph is being displayed will revert to the latest active time/temperature display above.

Note that pressing button 2 (or hitting "h" on the keyboard) during the graph display will capture the current temperature and time, but will not terminate the graph display. Instead, the current time/temperature display will appear the next time the graph is touched/clicked.

Also note that the graph may appear blank after a Restart, until enough data has been recorded to show a complete graph. (i.e. After enough recording time intervals have expired.)

Menu

If you press button 4 on the PiTFT (or hit "m" on the keyboard) either display above will switch to the following main menu display. Time and temperature will still be captured during the menu display and the updated graph or temperature display will be restored when you select the Return option.



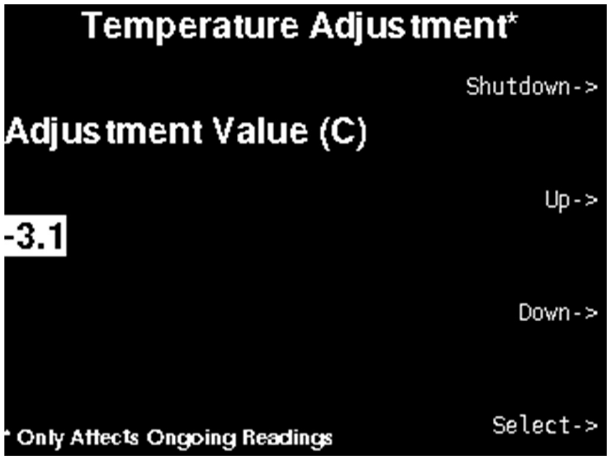
In menu mode, if a keyboard is available, the "u" key will simulate PiTFT button 2, to scroll the highlighting up through the options; the "d" key will simulate PiTFT button 3, to scroll down and the "s" key will simulate PiTFT button 4 to select the highlighted option. As before the "x" key, or clicking the upper-right "X" on the X11 window title bar will exit the software at any time in menu mode during testing.

The Return option will simply revert to the temperature display, while the Exit option will exit the programme completely.

Temperature Adjustment Menu

The Adjust Temperature menu allows a fixed temperature adjustment to be applied to all ongoing readings. You might use this if, for example, you notice that the temperature on the device does not match a more accurate temperature source that you have available. The value that is shown when this sub-menu appears for the first time reflects an adjustment value that is hard coded into the software.

The big assumption here is that the basic Type K thermocouple readings are linear enough within a practical usage range at home.



Up will increase the temperature adjustment by 0.1 degrees C and Down will decrease it by 0.1 degrees C within a limit of +/- 10 degrees C.

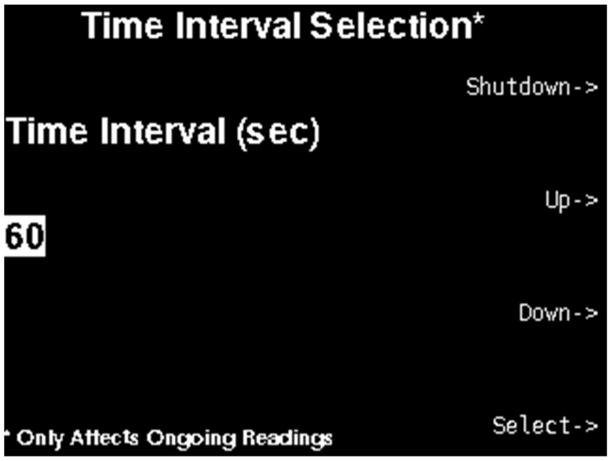
This will not affect already recorded readings - which could result in a change in the ongoing graph - until a value is Selected. In general it is a good idea to do a 'Restart' from the main temperature or graph display after making an adjustment change, to start with fresh and consistent temperature readings.

The chosen value will only take effect, once the Select button is pressed, so it is possible to undo a change by scrolling to the value that was shown when the menu initially appeared and Selecting it.

Note that adjustment value changes are not currently saved across shutdowns and so must be re-entered each time that the software is started.

Time Adjust Menu

The Time Adjustment sub-menu allows the time interval that is used for data collection (e.g. For graphing and for csv files) to be changed from a list of values that are hard-coded in the software.



The chosen value will only take effect, once the Select button is pressed, so it is possible to undo a change by scrolling back to the value that was shown when the menu initially appeared. Previous data that has been recorded will not be lost unless a Restart is performed.

The list of values that is currently available is 6, 30, 60, 120, 600, 1800 and 3600 seconds, with the default being 6 seconds, so that you can watch the graph change quickly the first time you start it. The largest values may be useful for measuring things like refrigerators or outdoor temperatures.

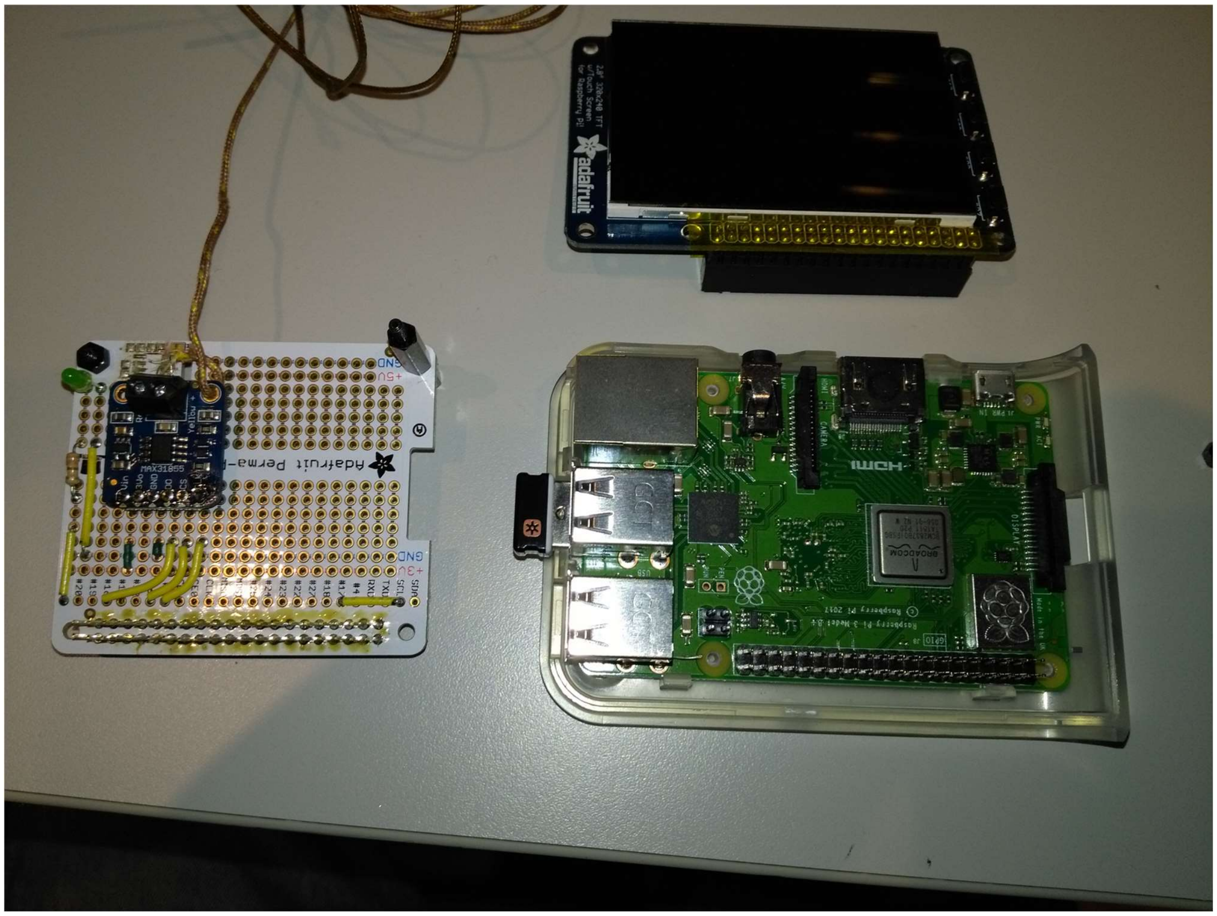
The Up button will move through increasing values from the available values and the Down button will move through the decreasing values.

Hardware Overview

The modestly named World's Greatest Oven Thermometer (WGOT) consists of three, stacked subassmblies.

1. A Raspberry Pi3 B bottom board
2. An Adafruit Perma-Proto Hat with MAX31855 board & connections - middle board
3. An Adafruit 2.8" PiTFT capacitive touchscreen, with buttons, on top

The following photo shows the prototype subassemblies.


Note: This shows a Hammond base case, but the Adafruit one would work just as well.

Hardware Assembly

The following is a list of the parts required to build the prototype shown – just a starting point – using Adafruit part numbers.

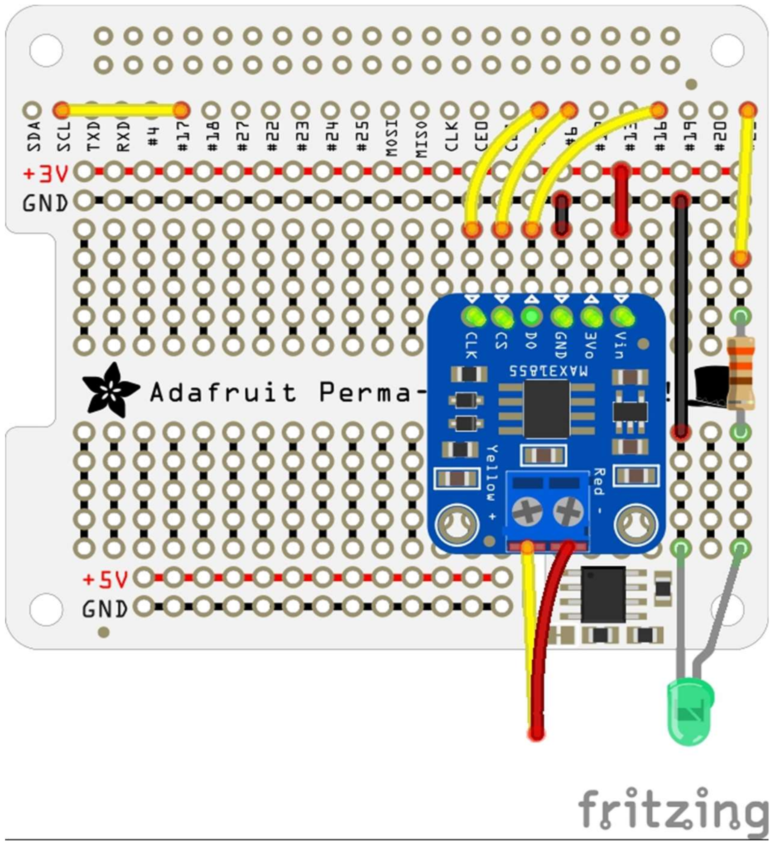
Note: In Canada, most Adafruit parts are available from Elmwood Electronics, to avoid shipping delays and customs, eh.

Sub-assy	**Description**	**Adafruit Part #**
CPU		
	Raspberry Pi 3 B+	3775
	Power Supply	1995
	32 GB micro SD card	
Sensor		
	Thermocouple (S/S tip)	3245
	MAX31855 board	269
	Perma-Proto Pi Hat (no EEPROM)	2310
	stacking header – extra long pins	2223
	LED, bright green	
	330 ohm Resistor, 1/4 watt, 10%	
Display		
	2.8" PiTFT Cap. Touchscreen	2423

Hardware		
	M2.5 11mm normal standoffs(2)	2336
	M2.5 16mm tall standoffs(1 of 2)	2337
	base case – clear (optional - see Enclosure Design below)	2253

The standoffs are important, so as to avoid stressing the PiTFT board when pressing its buttons. The headers themselves provide support to the other side for all boards.

The following image shows the wiring for the Perma-Proto board with the MAX31855 board attached.



Note that there is enough space in the slot below the MAX31855 board through which the thermocouple wire can be routed, to provide some strain relief (instead of routing it through one of the MAX31855 mounting holes - which I did - at the cost of ripping off the shrink wrap in the process.)

One thing to note as well is that the MAX31855 board is positioned so that it does not interfere with the PiTFT board above it and so that it is far enough from the Pi CPU chip to avoid CPU heating from affecting its internal temperature calibration too much. The MAX31855 temperature output values do seem to be skewed if there is much of a temperature difference between where its chip is located and the position of the thermocouple connection. This positioning also ensures more room on the Perma-Proto board for more components, like a speaker or maybe a Bluetooth board.

The pins from the long pin header soldered to the Perma-Proto Pi Hat board extend high enough so that the 2.8" PiTFT screen can then be plugged onto them.

Technical Details

The following table shows the GPIO usage for all of the hardware components, using BCM numbering and not Raspberry Pi pin numbers.

Device	**GPIO#**	**Usage**	**Notes**
2.8" PiTFT Screen			Adafruit #2423
	2	SDA	I2C
	3	SCL	I2C + power on (short to GPIO 17)
	8	CE0	SPI0
	9	MISO	SPI0
	10	MOSI	SPI0
	11	SCLK	SPI0
	17	Button 1	Top button = power off
	22	Button 2	
	23	Button 3	
	24	PiTFT	
	25	PiTFT	
	27	Button 4	Bottom button
MAX31855 Board			S/W SPI
	5		CLK
	6		CS
	16		DO
Activity LED	21	GPIO.OUT	SPI1 SCLK
Reserved-ish			
	7	CE1	HAT ID?
	18	PWM0	Used for headphone jack?
Free-ish			
	0	ID_SD	
	1	ID_SC	
	4	GPCLK0	
	12	PWM0	
	13	PWM1	
	14	TXD	
	15	RXD	
	19	MISO	
	20	MOSI	
	26		

Pre-requisite Software Package Installation

The nice thing about open source software is that it's all free. Unfortunately, this also means that various packages change so fast over time that any development effort is out of date by the time that it is released – probably like for this one. This software is (now) based on Python 3 though it works with Python 2, so your mileage may vary.

e.g. Current pygame SDL support for TSLIB touchscreens in Stretch means that we can only detect touchscreen events, but the positition information and other TSLIB filtering are lost.

In this document, what is described below is based on Raspbian Stretch, 2018-06-27, so you may not experience the same results at home. (e.g. I am still having issues with Raspbian Stretch 2018-11-13 to force the display to the PiTFT.)

The following is what I did.

In addition to installing the full version of Raspbian Stretch, the following software packages need to be installed.

- Adafruit MAX31855 support
- Adafruit PiTFT support
- Matplotlib for Python

All other packages, like Pygame or RPi.GPIO are included in Stretch or derived from the above.

Adafruit MAX31855 Support

To install the Adafruit MAX31855 support, go to the following link to download and install it, following all of the instruction steps.

https://learn.adafruit.com/max31855-thermocouple-python-library/software

If you are planning to use Python 3, it might still be necessary to use the following commands to install the MAX31855 libraries for it, in case you get "import" errors, because Python 2 is/was the current default for the 'python' command with Stretch as of this writing.

```
cd ~/Adafruit_Python_MAX31855
sudo python3 setup.py install
```

Adafruit 2423 2.8" Capacitive Touchscreen Support

To install the Adafruit 2.8" Capacitive touchscreen support, go to the following link and use the Easy Install process provided. Watch for any errors and follow any advice provided if they occur.

https://learn.adafruit.com/adafruit-2-8-pitft-capacitive-touch/easy-install-2

Select your PiTFT device (3 for the PiTFT Capacitive Touchscreen, choose rotation option 1 (landscape) and then say "no" to the the next two configuration questions – so that the PiTFT can be used as a separate framebuffer device from the normal X11 console.

It also is a good idea to make sure that the PiTFT works as /dev/fb1 by following the instructions at in the following link.

https://learn.adafruit.com/adafruit-2-8-pitft-capacitive-touch/displaying-images

Note: Swiping the PiTFT screen while in X11 testing mode can cause unpredictable results on the X11 screen, even when the software is not running. So be careful if you want to clean it!

Matplotib

To install Matplotlib, go to the following link and follow the instructions for Linux.

https://matplotlib.org/users/installing.html

Be sure to use the apt-install process for Debian instead of the pip-install process that shows up first.

e.g. As of this writing, use the following command.

```
sudo apt-get install python3-matplotlib
```

You can also try the following command on your Raspberry Pi if you are going to use Python 2.

```
sudo apt-get install python-matplotlib
```

Other Packages

You may also want to enable VNC – already installed in Stretch - on your Pi, if you want to do remote development as well as to easily transfer files. Also, if you want to print from your Pi, you can install CUPS, though you'll likely have to figure out the closest printer that matches what you actually have.

OS Customization

There are two main OS customizations that are required for this project, if you want to use their associated features. These should be done after all pre-requisite software packages are installed.

1. Update /boot/config.txt so that PiTFT button 1 (labelled #17) will cause a shutdown.
2. Update /etc/rc.local so that the software is started at boot time – e.g. for "headless" mode.

Shutdown/Restart Button

To use button 1 on the PiTFT as an OS shutdown button, add the following lines to end of the /boot/config.txt file using, say, "sudo nano /boot/config.txt". Ideally, updates from the PiTFT display install should appear at the bottom of the file if you've used the Adafruit Easy Install script for the PiTFT already, which you should have done previously.

```
# --- Added for WGOT project

# --- Make GPIO 17 (button 1 on PiTFT 2.8") shutdown when pressed

dtoverlay=gpio-shutdown,gpio_pin=17

# --- end WGOT project updates
```

Note that using button 1 to also restart after shutdown is accomplished by connecting GPIO 17 to GPIO 3 on the Proto-Perma PiHat board. (This is OK, because the GPIO 3 I2C SCL pin can be pulled to ground – part of the I2C protocol.)

Auto Start at boot

When you are done testing and want to run the fully assembled device in headless mode, you need to update /etc/rc.local to automatically start the WGOT software.

Add the following lines to the end of rc.local, but before the "exit 0" statement using, say "sudo nano /etc/rc.local".

```
# Start WGOT Python program in separate process

sudo python /home/pi/Proto28.py &
```

Note that the above contains the full path name for the file that I used. Yours may be different, but it must be the full path name for the file. Also, the "&" is very important, to ensure that the progamme is started as a separate process and does not hold up the rest of the OS startup process.

Alternatively, if you want to log messages from stdout and stderr to shifting log files – so you don't lose the log data over reboots - you can try this version of /etc/rc.local.

```
#!/bin/sh
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.

# Print the IP address
```

```
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
  printf "My IP address is %s\n" "$_IP"
fi

# Shift WGOT log files, ignoring errors (ie no -e flag on the first line above)
# Target directory must be enabled for 'write' for 'others'
rm /home/pi/WGOTlog2.txt
mv /home/pi/WGOTlog1.txt /home/pi/WGOTlog2.txt
mv /home/pi/WGOTlog.txt /home/pi/WGOTlog1.txt

# Start WGOT Python program in separate process and log msgs
# The final ampersand makes it start in a separate process,
# so rc.local can complete
sudo python /home/pi/Proto28.py  >/home/pi/WGOTlog.txt 2>&1 &

exit 0
```

For the example above, which uses /home/pi to hold log files, you first have to let rc.local, which runs as root, to write to the home/pi directory - e.g. using the following overtly permissive command.

```
chmod 777 /home/pi
```

If you have made a terrible decision to go headless and the software bombs, it is not the end of the world. You can always reconnect your HDMI monitor and keyboard and restart the Pi, so that you can update /etc/rc.local to comment-out the auto-start command while you fix the problem.

Software Overview

The software is heavily commented, because,well, comments are good! Though not pretty, the software might also suggest some answers about Python, Pygame, Matplotlib, SDL, PiTFT and other questions for which I couldn't always find useful answers on various forums.

Software Testing/Debugging

The software currently offers some testing variables that are defined near the top of the code for ease of access.

1. Forcetft, which if set to True might force the display to the PiTFT, when started from an X11 window. If set to False, it will allow SDL to choose the display target – usually a window on an X11 display. (Doesn't work with Stretch 2018-11-13 as of this writing.)
2. Debugprt, which if set True will print some debugging info, for testing purposes or, if set to False, will supress these messages. There are also some debug print lines that are commented out, because they are annoying – except for very detailed debugging.
3. Glitchless, which if set True will smooth out some of the one time temperature glitches I kept experiencing.
4. Tempadj, which is a fixed +/- Celsius temperature value to be added to the readings from the MAX31855 based on your calibration testing.
5. Csvfilename, which is the full path name for the file to hold recorded values in csv format.
    - This is set to '/home/pi/WGOTdata.csv'
    - Saved csv files use the name above with save1 or save2 appended to the name.
6. Graphfile, which is the full path name for a temporary file to hold graphs.
    - This is set to 'home/pi/WGOTgraph.png'

First try the following command to see what happened when rc.local was run.

```
systemctl status rc.local.service
```

In addition, if the software is started in headless mode and something goes wrong, you can try looking at the following verbose sources for error messages.

/var/log/boot.log

/var/log/messsages

/var/log/daemon.log

You can also modify the commands in /etc/rc.local to direct stdprint and stderr output to a log file of your choosing, as shown in the example above.

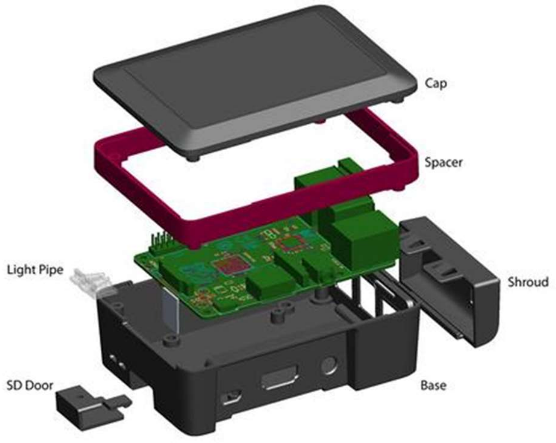
Enclosure Design

The following example is for those who would like a neat, self-contained enclosure, but with a minimum of mechanical skills, unlike, say, Ben Heck.

It consists of a Cyntech Raspberry Pi case, - with a spacer to accomodate the extra height of the Perma-Proto Hat – and a modified Adafruit PiTFT faceplate with buttons (Adafruit #2807), sandwiched between the cap and the PiTFT. There is also a handy hole on the side of the Cyntech case for the thermocouple cable to exit the case. I also chose the clear Cyntech case so that the activity LED would be visible from above – reflected through the case itself.

I also passed the thermocouple probe through an alligator clip, so that I could attach it to an oven grill or fridge grill.

The following diagram shows how the Cyntech case works – copied from the Cyntech web site.

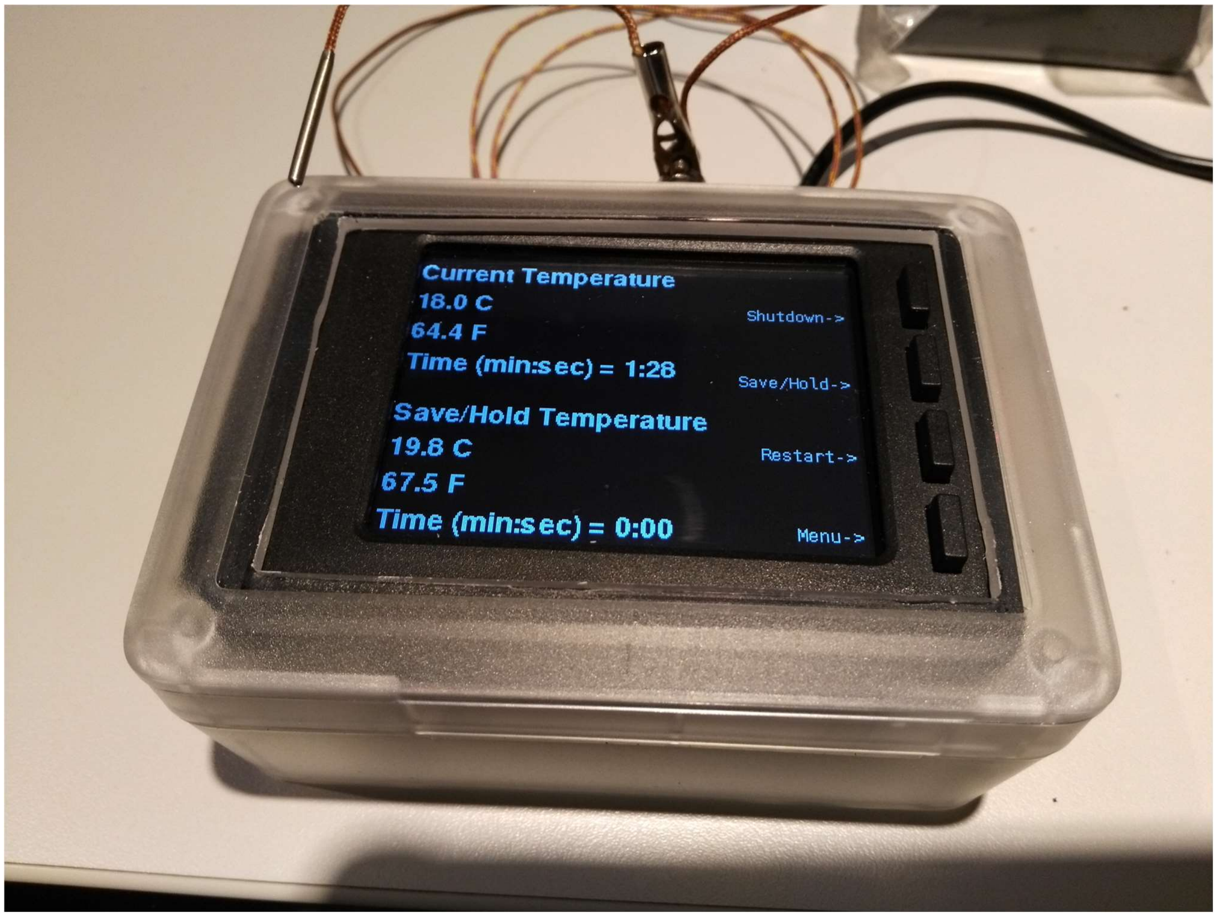


Modification to the Cyntech case requires milling out the shiny part of the cap to expose the screen and buttons of the Adafruit PiTFT faceplate that will go just below it. I used a Dremel tool and a file to do this part.

Modification to the Adafruit PiTFT faceplate requires cutting out the corners on the long side of the faceplate to clear the screw holes for the Cyntech case, as well as bevelling both ends to better mesh with the Cyntech cap – it's a tight fit. Once again, I used a Dremel tool and a file, though a fine saw and a file might be better.

Note that is best not to tighten the assembly screws too much or the buttons on the faceplate may not have enough space to move properly.

The final assembly is show in the following picture.



Future Improvements

So, what's not to like?

Well, there could be just a *few* improvements - such as the following – in the fullness of time.

- Change the name. (to World's Greatest Appliance Thermometer – WGAT. :-)
- Accept command line arguments for some variables.
- Save the time and temperature adjustment values over shutdowns.
- Save csv log data to a USB memory key.
- Add more menus for setting things, like C vs. F display/graph, saving to USB, smoothing, etc.
- Build a more general purpose menu tree driver without making the code any harder to read.
- Figure out the glitchy temperature readings.
- Better debugging tools, cleaner, easier to read code, etc.
- Much, much more.

In the interest of sharing, and before the software balloons too large to comprehend, I am publishing this prototype version.so that newbies like me may benefit from some of the lessons I've learned.

A Big Thank You to Adafruit & friends for their products, software, documentation, tutorials and forums! As a newbie to Python, Pygame, etc., this wouldn't exist otherwise.