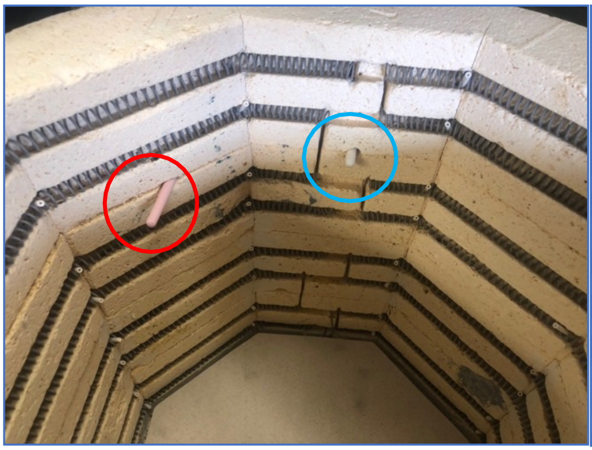
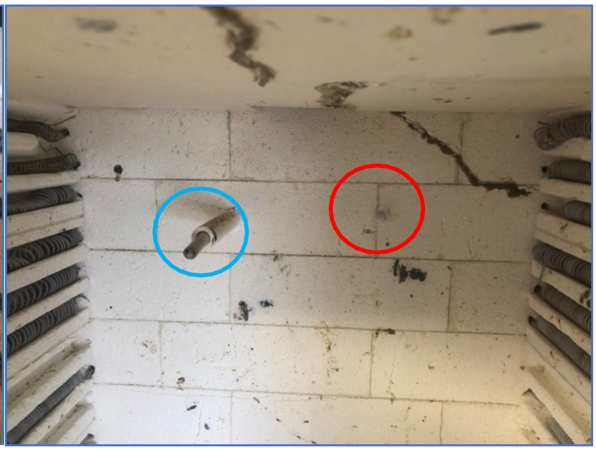
Mark Tilles' iteration of Jason Bruce's kiln controller project
updated Monday, August 1, 2022

My ceramic workshop has three kilns in the adjoining garage, two 60 liter Rhodes and an old 120 liter Chematex. One of the Rhodes had a more-or- less modern Bentrup TC-40 digital controller, although with only one curve at a time you had to change for each firing need. The Chematex had a very old analog "Keramikexperterna" controller with a bunch of LEDs (basically junk). New controllers made for these kilns were very expensing I had noticed, but since I am mechanically inclined and computer literate, I decided I'd start looking for alternatives.

At first I wanted to evaluate the actual accuracy of the controllers that came with the ovens, so I ordered a pair of inexpensive REX-100 PID-type controllers and a pair of S-type thermocouples from Wish and Banggood. I installed second thermocouples (**red**) into each of the ovens and wired them up to the new PIDs, then proceeded to compare the temperatures these parallel systems reported compared to the original controllers still attached to the ovens.



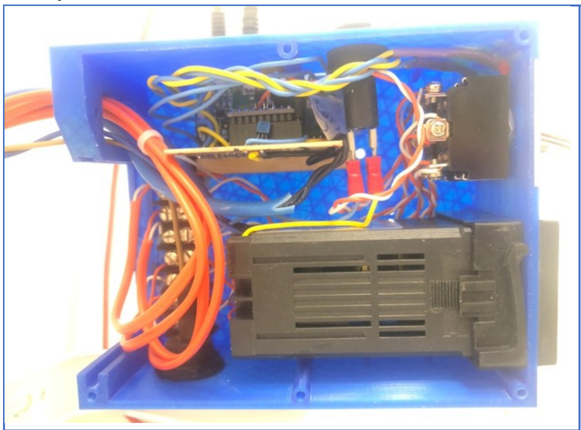
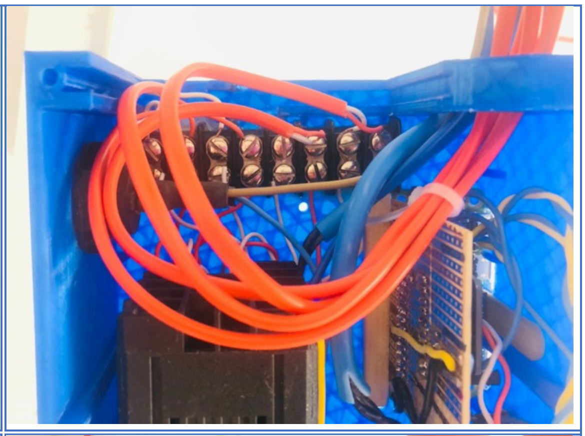
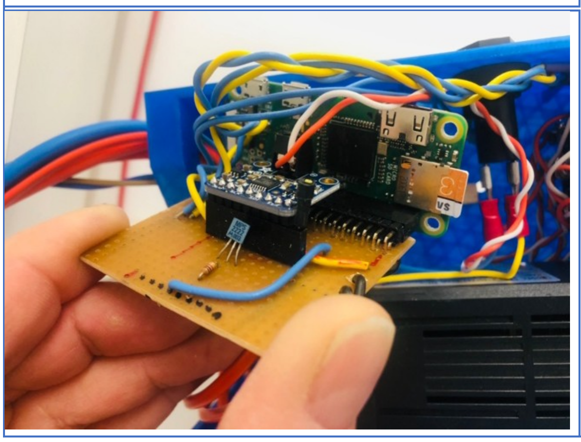
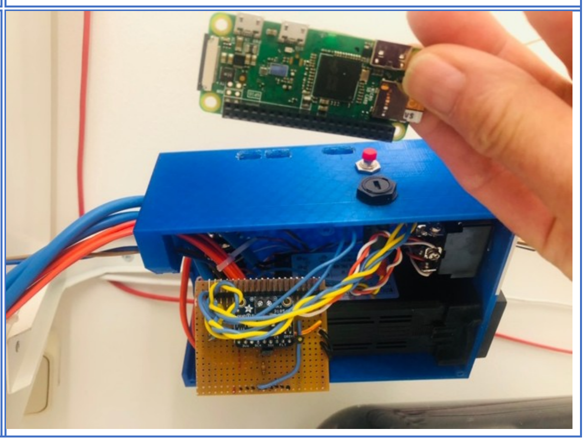


What I discovered was that (LOL, not surprisingly) my new equipment from China wasn't exactly high quality, and temperature measurements didn't match too closely either between the two PIDs, nor the PIDS to the original oven controllers. First I thought I could use these PID controllers to actually control the ovens, but I soon learned that there were also "firing curves" involved in ceramics, and I'd need to spend "real money" to find PIDs that were capable of these. One further complication I didn't understand was the need to use actual "thermocouple" wire when connecting the thermocouples to the controllers, but this came in my next stage of development.

So, I started searching for projects on the Internet and discovered Jason's project, which I proceeded to get rolling with a Raspberry Pi Zero W. Early on, Jason's code only supported the MAX31855 interface, which only supports K-type thermocouples; but he gladly assisted me in figuring out how to get the MAX31856 interface card working, since my ovens already had S-type thermocouples and I didn't want to change them. Finally, after a few weeks of late night "hacking" with Jason and Google as my friends, I had a working system and Jason updated his material as needed.

Since I have only one 3-phase 16A electrical socket in my garage, I wanted to create one programmable kiln controller control system that either of two kilns could use – and *easily*.

Over a period of a few weeks I had numerous in-depth conversations with an expert in the field of thermocouples, and his position was that every connection in the path from the thermocouple to the controller card would create a voltage differential, and thereby negatively affect the accuracy and operation of the system. I was warned that unless I minimized the number of connections as well as used proper thermocouple wire, I'd likely have either operational problems or unstable results. Nevertheless, I used my 3D printer to print a small controller box to hold my hardware, cut up a piece of parallel-trace circuit for the necessary wiring and SSR feed transistor on, and used connectors so both the MAX31856 and pi zero units were both easily replaceable. I then installed DPDT switches into the box that I had ordered on Banggood with which I could switch the kiln connected to the pi controller.

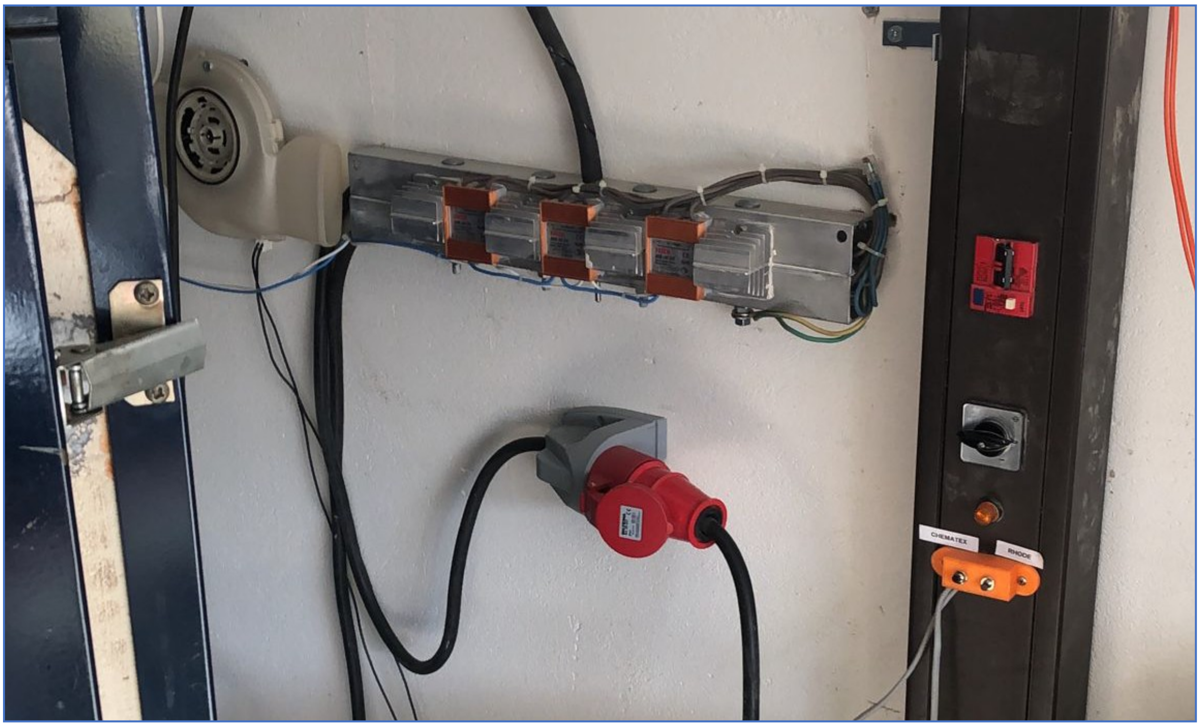





Yes, this added a lot of connection points, a “no-no” in the field of thermocouples, but I did in fact use thermocouple wire for S-type thermocouples ordered from RS-Components in England, ***RS PRO Thermocouple & Extension Wire 25m article: 611-7902***. And guess what? In the end this system ended up working like magic, as I am getting reliable and consistent results every time with either oven. Not a single misfiring in over two years due to Pi- or thermocouple trouble. Now, keep in mind the controller is NOT in the often-hot garage, but in the adjacent workshop which is always at a comfortable indoor temperature. But let me continue …

Here is a picture of my first finished controller box (I now have two, one as a backup and for development). What I will point out first is that the PID controller seen mounted on the left is an *entirely separate control system wired in place only for overheat protection* – it has no physical connections to the pi system - more to come on this later.

OK, so now I had two ovens with two thermocouples each, and two DPDT switches in my controller box that flipped the MAX31856 interface card and PID controller between the ovens. That’s great, *but now I had to figure out the actual working of the 3-phase control and powering of the ovens.*

Understanding that it was best to use solid state relays for this application, I ordered a bunch of 240VAC/40A single-phase solid state relays from Banggood (very cheap). Upon disassembly of one of them I discovered it was true what I had read on the Internet: that these name-brand clones actually only contained 30A components in cases labeled as 40A, but I didn’t care as I only needed 16A max. ©. I next built a big heat sink from square extruded aluminum tubing I had lying around from window awnings I had torn off the house. I mounted the relays and some cheap aluminum heat sink towers to it as well, and to ensure the relays stayed cool, I mounted an old dishwasher fan I had lying around beside it for additional cooling through the tubes, if needed. I used the same heavy-gauge cord for this “power brick” as the ovens had, and all three SSRs were coupled in parallel so the pi could fire them all at once. I also printed SSR covers on my 3D printer to cover their power connections safely. **Update**: I have mounted a new 3-phase contactor on the cement wall that has an “enabling switch” connected in a current loop through the oven magnetic door sensors of both ovens, so that power to the SSR power bank is only enabled if the oven doors are shut and the button pushed once. If a door is opened the button must be pushed again. I wanted this in case of power fluctuations or spikes that might cause the Pi to lock up … I want to be required to inspect the system before the oven power is re-enabled.



Now, the pi controller manages this one power brick, and with the 3-phase wall jack I also mounted onto the cement wall, I could select which oven I plug in at any time. **Not shown**: The 3-phase contactor now installed.

Next, I ripped out the original control circuitry from each of the ovens, all except the heating elements. But this had me worried a bit. I had read in forums that when SSRs fail, they *usually fail stuck in the on* status. This means I could have a thermal runaway event, potentially dangerous and costly. What to do? Again, the solution was relatively simple. I would create an "overheat protection circuit" *completely independent from the pi system* that would permanently shut down the magnetic relays. It works like this:

The REX PID controller you saw in the picture on the previous page is also switched between the ovens with its own DPDT switch, connected to the second thermocouple installed in each. This PID has its external relay contacts wired to its own SSR, which in turn enables or disables 230VAC to the above mentioned physical contactor. As long as the PID sensor temperature is under its alarm setting (currently 1270C) the contactor would be energized and thus the SSR power brick enabled. If this temperature were ever exceeded, the PID would shut down the SSR powering the magnetic relay and the oven would shut down. But this wasn't foolproof, as it would only prevent the temperature from *continuing* to rise further, because as soon as the temp dropped below 1270 it would again turn on the contactor. I needed to make this safety system shut down the whole process. How to solve?

Most 3-phase contactors have an extra, unused set of terminals for "control" applications. I simply wired the extra 4th pair on contacts on my relays into a "power loop", together with a SPST pushbutton which momentarily energize the relay, and this then closed that extra set of contacts on the relay which then energized itself, so it would stay energized until it lost power. Then it would require a physical push of the button again to re-energize. I have this button also mounted on the cement wall. Unless the button is pushed, the SSR power brick cold never receive power.

Now I have a fully functional, physical control system in place, with an independent safety cut-off system, but the kiln- controller project was only designed to operate one oven. Yet I have two ovens.

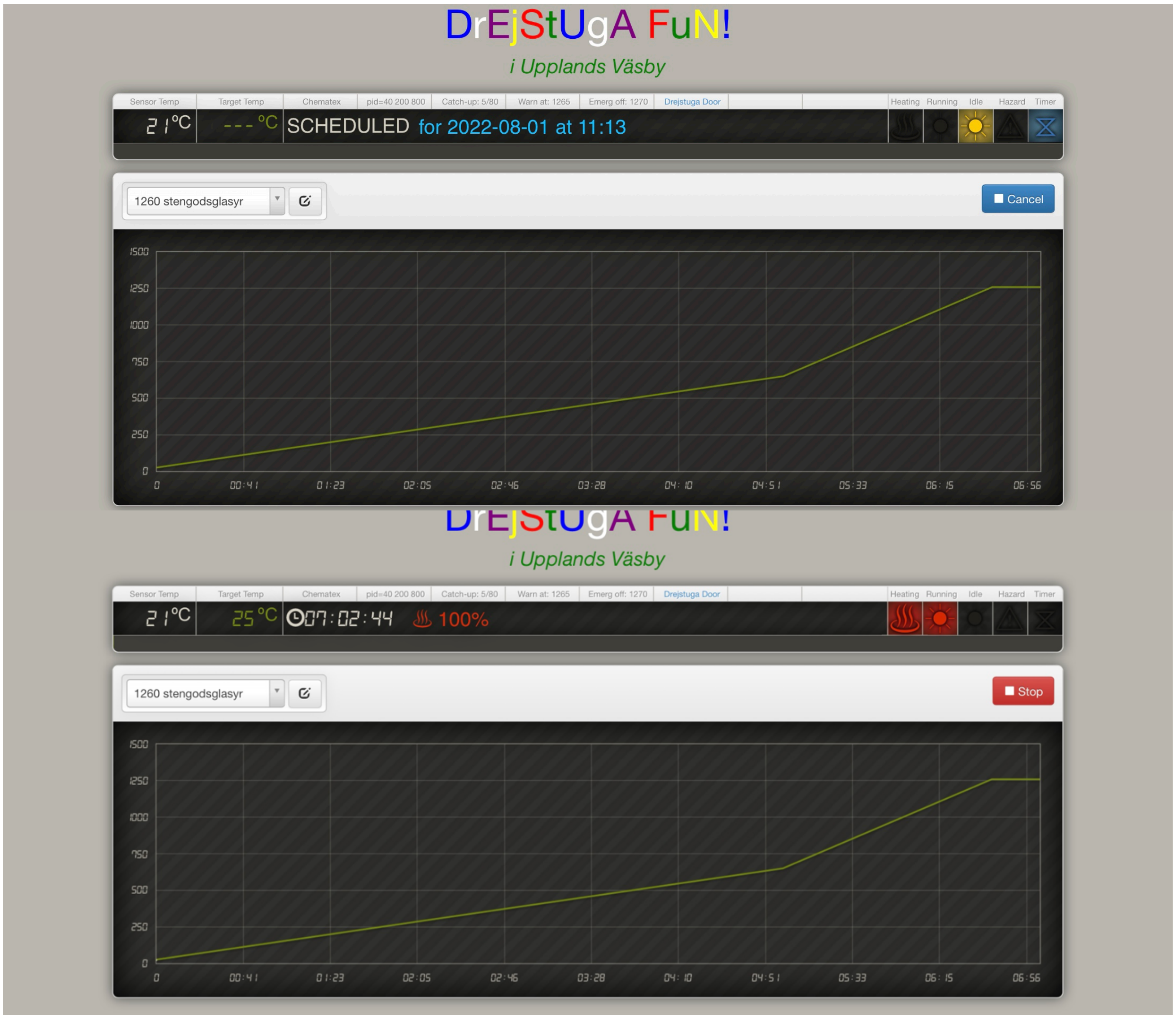
Sure, I can plug either oven into my power block, but since every oven has different PID parameters I would have to change the parameters and restart the kiln-controller subsystem each time I changed ovens, and that would be a pain. What to do? The next solution got tricky, and I'm guessing that none of you reading this document will be using the controller with multiple ovens. Nevertheless, I will explain the system modifications I have made to the kiln-controller software to accomplish this.

I first thought about having multiple config.py files, one for each oven, and just copying in one or the other and restarting the service. But this would still a hassle since I'd have to SSH into the pi to do this. There is also the curl api that could maybe be used, but again – command line stuff. What I wanted was a way to use the web interface to swap between the oven configurations, and that's what I did.

Over time I have heavily tweaked the web interface and added backend system commands in the python and javascript code, many of the additions and changes are listed below.

In the index.html file you will notice not only different icons, but also there are new functions and variables displayed. I like being able to view a lot of system status information before and during a firing, and this is what I've done.

The following is a screenshot of control.html and notes (and I will be editing these soon, the above text is all I have the energy to document today!)



I have made changes to the web page title, header text, icons, and functions etc for my application. I also added a clickable javascript link over the kiln name (available only during state=IDLE) which causes the kiln- controller system to run underlying system scripts to swap between kiln-controller instances in parallel folder paths.

So I now have a dual-oven setup, with physical switches swapping the thermocouple connections between ovens. This stuff isn't likely of interest to anyone but me. But other changes and enhancements you might want to look at include:

1. Changed the icon titles to: **Heating Running Idle Hazard Timer**

2. Improved the icons under the titles. Changed some standard background colors (although I used existing css, didn't make new css, so the css format names don't exactly matching where they are being used)

3. Enabled dual color for the Heating icon: if heating at 100%, red; if heating 0% <heating> 100%, yellow.

4. When firing curve is not running, the Idle icon lights up.

5. When a firing curve is running, the Running icon lights up.

6. When a start-delay timer has been enabled, the last icon “Timer” will be blinking.

7. Fixed the hazard function, as the variable never seemed to reach the picoreflow.js. Now, when the sensor temp reaches within 5 degrees of the shutdown temp, the hazard icon will light. See "Warn at:" info below.

8. Added new fields in the info line:

   Kiln: “kiln name” - kiln name from config.py

   pid=20 70 200 - "pid=" - the pid values from config.py

   Catch-up: 5/60 - "Catch-up:" - the catch up value, if enabled; otherwise "off". Plus, I added a new "ignore" temp below which overheating swings will be ignored by catch-up function. This avoids long and unnecessary cool-down pauses when the firing curve start temp is higher than the oven temperature (for example, my garage here in Stockholm can get very cold in the winter, and this causes an overshoot at the start of the firing unless I change the curves to start very near the oven sensor temperature. But maybe I can continue to pid-tune this out? Anyway, this new ability to ignore overshoots at low temps comes in handy for me.

   Warn at: 1265 - "Warn at:" – a hard-coded value of the emergency shutdown value from config.py - 5 degrees. At this temp the "Hazard" icon will light. Maybe add email hooks later for extra warning?

   Emerg off: 1270 - "Emerg off:" - the emergency shutdown value from config.py

9. **Heating %:** - is the live % heating capacity shown on the main status line. This is the logged "pid" value (2.00 max = 100%). To do this I fixed the value of "heat" in oven.py so it reaches picoreflow in its entirety; it was previously being swapped out for 0 and 1 and I don't know why since it doesn't seem it would affect system operation by doing this. It was sent in during simulation firing in its entirety though ... I hope to later have an average value over the previous minute instead, which would be more representative of the heating capacity the oven is working at.

10. Fixed the kwh cost estimate, as the kwh cost value was hard-coded and not reaching picoreflow.js

A few other notes regarding installation:

- I have added a few extra informational LEDs into my system, and they are controlled using gpiozero. So unless you remark out these first eight lines of kiln-controller.py, you will have problems starting up unless you also install the RPi.GPIO module.

```
# ADDED BY MARK TILLES TO START BLINKING GREEN LED WHEN SERVICE IS RUNNING
from gpiozero import Button, LEDBoard
from signal import pause
import warnings, os, sys
green_ledGPIO = 6
green_led=LEDBoard(green_ledGPIO)
green_led.blink(on_time=1, off_time=1)
# END – ADDED BY MARK TILLES TO START BLINKING GREEN LED WHEN SERVICE IS
RUNNING
```

- More notes to follow … too tired to continue just now!