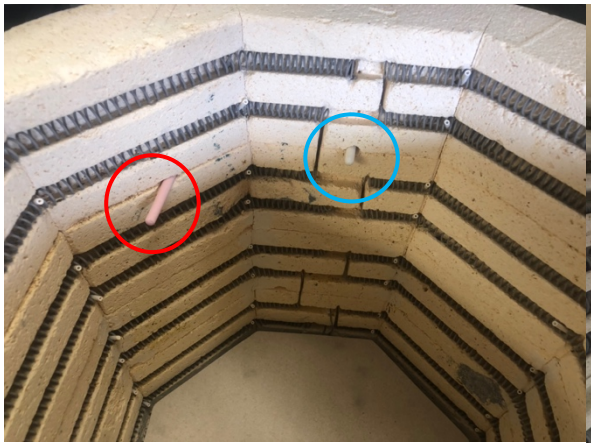
Mark Tilles' iteration of Jason Bruce's kiln controller project
updated Friday, October 15, 2021

My ceramic workshop has two kilns in the adjoining garage, a 60 liter Rhode and an old 120 liter Chematex. The Rhode 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 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, knowing I still wasn't there, I started searching for projects on the Internet and discovered Jason's version of the picoreflow controller, which I proceeded to get rolling with. 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 sensor system and Jason updated his material as needed.

Because I have two kilns, yet only one 3-phase 16A electrical socket in my garage, I wanted to create one kiln controller control system that either kiln could use – and *easily*. During this time I had numerous in-depth conversations with an expert in the field of thermocouples, and the consensus 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.

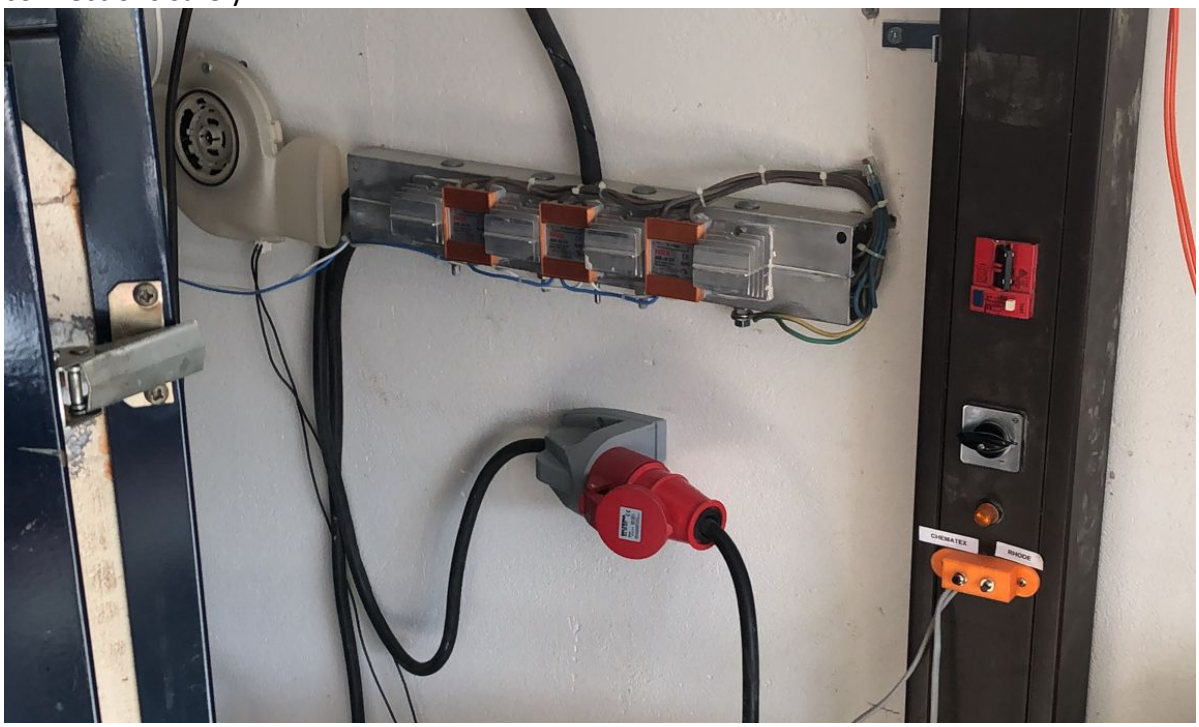
Nevertheless, I used my 3D printer to print a small controller box to hold my hardware. I then installed DPDT switches into the box that I had ordered on Banggood with which I could switch the kiln connected to the pi controller. Yes, this added a lot of connection points, a "no-no" in the field of thermocouples, but I did in fact use thermocouple wire for S-type thermocouples I ordered from RS-Components in England, ***RS PRO Thermocouple & Extension Wire 25m article: 611-7902***. And guess what? In the end this ended up working like magic, and I am getting reliable and consistent results every time with either oven. But let me continue …

Here is a picture of the finished controller box. What I will point out first is that the PID controller seen mounted in the same box as the Raspberry pi is an *entirely separate control system* I have in place *only for overheat protection* – it has no physical connection to the pi - more to come on this later in this document.

OK, so now I had two ovens with two thermocouples each, and DPDT switches in my controller box that flipped the MAX31856 interface card between the original thermocouples in one or the other oven. That's great, *but now I had to figure out the actual powering and control of the power to 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 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.



Now the pi controller manages this one power brick, but with the 3-phase wall jack I mounted to its output I could select which oven I plug in at any time.

Next, I ripped out the original control circuitry from each of the ovens, so that they would simply receive their power directly from my power brick into their magnetic relays. All I would need to do would be to trigger those magnetic relays on during operation.

But this had me worried a bit. I had read in the 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? Simple! I would create an "overheat protection circuit" completely independent from the pi system. It works like this:

The REX PID controller you saw in the picture on the previous page is also connected to one oven or the other, depending on which position its own DPDT switch is positioned. This PID is wired to its own SSR, which in turn provides or disables power to the magnetic relays in the ovens. As long as the temperature is under the PID alarm setting (currently 1270C) the magnetic relays would be energized and thus the ovens could heat. If this temperature were ever exceeded, the PID would shut down the SSR providing the magnetic relay power and the oven would shut down. But this wasn't foolproof, as it would only prevent the temperature from continuing to rise, I needed to make this safety system shut down the whole process. How to solve? Also simple!

The magnetic relays themselves often have a 4th set of terminals, used for control applications. I simply wired this 4th pair into its own power loop, together with the small SPST pushbuttons seen in the photo. Providing the safety system is powered on, the pushbutton would momentarily close the energizing power loop to the relay in the oven which allowed the relay to energize, and stay energized, until it loses power. Then it would require a physical push of the button again to re-energize. I have one button for each oven mounted on the wall. Unless its button is pushed, the oven cannot operate.

For further safety, I included the door- and lid magnetic switches in this power control loop, so that would also shut down the system.

Now I have a fully functional, physical control system in place, but the kiln-controller system was only designed to operate **one** oven.

But I have two ovens. Sure, I can pug either one into the controller, 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. That would be a pain.

What to do? That solution got tricky for me, and I'm guessing that few of you reading this document will be using the controller with multiple ovens, but I will nevertheless explain the kiln controller system modifications I have made 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. I have also renamed "index.html" to "control.html" and removed all control features from index.html. I did this so that I could have a web browser up on the clay room computer monitoring the firing, *without anyone being able to accidentally shut it down*.

In the control.html file you will not only notice different symbols used on the status line, but also there are new functions and many new variables displayed. I wanted to be able to see a lot of system status information before and during a firing, and this is what I've done.

Here are some notes (and I will be editing these soon, the above text is all I have the energy to document today!)

- Made changes to the web page title, header text, icons, and functions etc for myapplication. Also added a clickable javascript link over the kiln name which causes the kiln-controller system to run underlying scripts to swap between kiln-controller

instances in parallel folder paths. I have a dual-oven setup, with physical switches swapping the thermocouple connections between ovens. So that stuff wouldn't likely really be 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: Rhode - "Kiln:" - 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 axample, 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.

   Heating %: 0 - "Heating %" - the live % heating capacity. 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 ...

   Warn at: 1265 - "Warn at:" - 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

Waiting to: 13:20 - (Hidden from view, unless a start-delay timer has been enabled. This new start-delay function has been implemented int eh the foreground javascript and is thus dependent upon the web browser not losing its connection to the server – i.e. by a closing of the browser window, an energy saver, refreshing the screen etc. My friend will be moving this feature into the python backend in the future so it will be browser-independent later).

A bit more about the timer: Although this timer is dependent upon the browser staying active and connected to the server, it seems to work well. After clicking "Start" and then "Yes, start the run" on your selected firing curve you will be prompted for a delay time in **hours**. You can enter decimal values, for example a half hour is 0.5 … if you want immediate start, just leave the default to 0. After you click OK the status line will display the "Waiting for: (time)" when the firing curve will begin and the blue "Timer" icon will start blinking. Once the time has been reached, the normal operation of the oven controller will begin.

9. 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!