IOWA STATE UNIVERSITY
Department of Chemical and Biological Engineering

ChE 490H Capstone Project
April 27, 2016

Andrew Hughes, Advisor: Dr. Cory Stiehl

Oven Control System for Composite Testing

Introduction

The oven used by Team PrISUm for curing composite test samples was controlled by a complicated system that lacked several features desired by the team. The old system used an on-off controller on an Arduino that was programmed by a Raspberry Pi. It was difficult to use and had stability issues.

A new system was designed to add a web interface, data logging, and a PID controller with adjustable gain scheduling. Python 3.4 and the Flask web framework were used to develop the system, and the code was run on a Raspberry Pi.

Theory

Equations 1 and 2 show the parallel form of the PID controller.

$$e(t) = SP(\mathrm{t}) - m(t) \tag{1}$$

$$out(t) = k_p * e(t) + k_i * \int_0^t e(t) + k_d * \frac{d}{dt} e(t) \tag{2}$$

e: Error; SP: Setpoint; m: Measurement; out: Output; k_p, k_i, and k_d: Gains

Digital instruments measure in discrete time steps. Equation 3 is the PID controller for a step of 1 second.

$$out(t) = k_p * e(t) + k_i * \textstyle\sum_{i=0}^{t} e(t) + k_d * \big(e(t) - e(t-1)\big) \tag{3}$$

Equation 4 eliminates a concept called integral windup by limiting the accumulated error term ($intE$). The gain is added into the error before summation to eliminate large kicks when the integral gain is changed. This replaces the summation term in Equation 3.

$$intE(t) = \begin{cases} out_{max}, & k_i * e(t) + intE(t-1) > out_{max} \\ out_{min}, & k_i * e(t) + intE(t-1) < out_{min} \\ k_i * e(t) + intE(t-1), & otherwise \end{cases} \tag{4}$$

When the setpoint is changed, the change in error spikes. By measuring the change in measured temperature, creating Equation 5, the kick in the derivative term is eliminated.

$$out(t) = k_p * e(t) + \mathrm{intE(t)} + k_d * (m(t) - m(t-1)) \tag{5}$$

The output from Equation 5 is bound to equipment limits by Equation 6.

$$out_{\mathrm{lim}}(t) = \begin{cases} out_{max}, & out(t) > out_{max} \\ out_{min}, & out(t) < out_{min} \\ out(t), & otherwise \end{cases} \tag{6}$$

To eliminate noise in the measured temperature signal, the last n raw temperature measurements (m_r) are averaged using Equation 7.

$$m(t) = \sum_{i=0}^{n-1} \frac{m_r(t-i)}{n} \tag{7}$$

Software

PID Controller

```
111  err = setPoint - currentTemp

131  intErr += ki*err

134  if intErr > outMax:
135      intErr = outMax
136  elif intErr < outMin:
137      intErr = outMin

140  din = currentTemp - tempOld

143  output = kp*err + intErr - kd*din

146  if output > outMax:
147      output = outMax
148  elif output < outMin:
149      output = outMin
150  relay.ChangeDutyCycle(output)
```

Gain Scheduling

```
114  for n in range(len(tuneParams)):
115      if err > tuneParams[0][n]:
116          kp = tuneParams[1][n]
117          ki = tuneParams[2][n]
118          kd = tuneParams[3][n]
119          if kset != n:
120              if kset < n:
121                  intErr = 0
122              kset = n
123          break
124  else:
125      kp = 0
126      ki = 0
127      kd = 0
128      intErr = 0
```

Temperature Moving Average

```
49   aveTempL = deque([],10)

109  aveTempL.append(thermocouple.get())
110  currentTemp = mean(aveTempL)
```

Data Logging

```
165  logFile = open('/home/pi/OvenProject/data/logfile.csv', 'a')
166  logwriter = csv.writer(logFile, dialect='excel',
     quoting=csv.QUOTE_MINIMAL)
167  logwriter.writerow(['Time', 'SP', 'Temp', 'Output'])
168  logwriter.writerow([logTime, setPoint, currentTemp, output])

190  logFile.close()
```

Hardware

A custom board (shown in Figure 1) was designed to connect the Raspberry Pi to the relay, and the MAX31855 thermocouple converter. Provisions were also made to connect a character lcd, but they were never used.

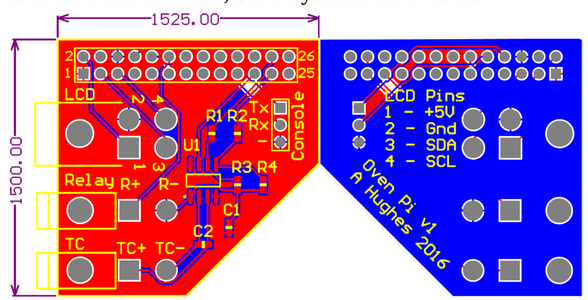

Figure 1: The final board shown by Altium Designer, a PCB design software.

During submission to manufacturing, the +3.3V and ground planes were omitted from the board, requiring extra wiring to maintain functionality. Figure 2 shows the finished board.

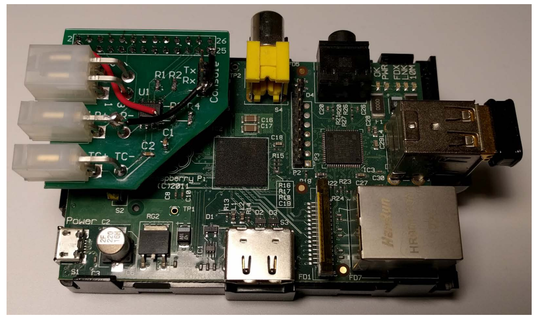

Figure 2: The assembled board mounted to the Raspberry Pi.

Results

The code and hardware were combined and deployed to Team PrISUm's oven. Figure 3 depicts the first test run on production hardware, before tuning. Figure 4 shows a test run after the gain schedule was tuned.

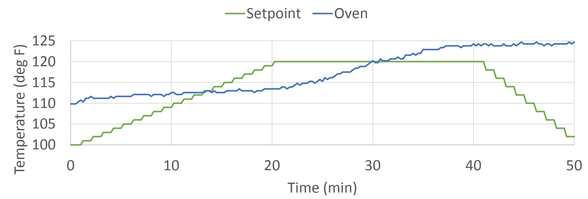

Figure 3: Heating to 120°F with untuned loop.

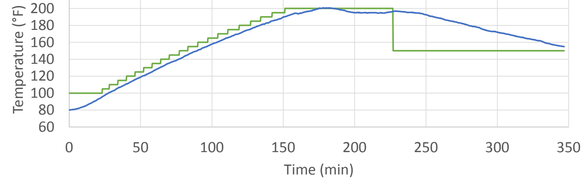

Figure 4: Heating to 200°F using the tuned gain schedule.

The large overshoot seen in Figure 3 is similar to the performance of the old system, and is almost completely eliminated with the tuning of the new system. The increased performance, combined with the new features, provided the team with a much more useful product.

All code and hardware designs are available online.
https://github.com/amhughes/ovenproject

References

[1] B. Beauregard, "Improving the Beginner's PID." [Online]. Available: http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-introduction/. [Accessed: 12-Oct-2015].

[2] Python Software Foundation, "Python 3.4 Documentation." [Online]. Available: https://docs.python.org/3.4/. [Accessed: 23-Sep-2015].

[3] J. Witucki, "Raspberry Pi driver for MAX31855." [Online]. Available: https://github.com/Tuckie/max31855. [Accessed: 21-Sep-2015].

[4] A. Ronacher, "Flask Documentation (0.10)." [Online]. Available: http://flask.pocoo.org/docs/0.10/. [Accessed: 20-Oct-2015].

[5] Philip Howard, "Raspberry Pi GPIO Pinout." [Online]. Available: http://pi.gadgetoid.com/pinout. [Accessed: 23-Sep-2015].

Acknowledgements: Team PrISUm