Cameron Givler

MIST Group

November 27, 2015

Laser Temperature Controller

Python/Arduino-based Temperature and Humidity Regulator

Introduction

The lasers used for quantum operations must be temperature-regulated to maintain their frequency. The ideal temperature range for these lasers is between 0 and 10 degrees Celsius. However, when temperatures are regulated to those relatively low temperatures, an issue that arises is dew forming on the laser. Although there are humidity controllers in the cooling system, these need to be regulated and if any issue arises, it must be addressed immediately. A malfunction that causes excess humidity will interfere with the laser and fixing the issue could take days. Therefore, the system must be flexible to alert the user and turn off the cooler when the humidity is approaching the dew point so action can be taken to protect the laser. The cooler currently in use for the main laser is old and difficult to precisely control. An upgraded cooling system will replace the one currently in use and add features that were not possible with the old system.

Objective

The objective is to build a new laser cooling and humidity detection system with an easy-to-use computer interface. This system must not only control the temperature, but also monitor the humidity and turn off the cooler when the humidity gets too high. In addition, 2 hours of temperature and humidity data must be stored and graphed in the user interface, and must be available for export, in the case of an anomaly. Finally, when the temperature is approaching an unstable value or the humidity rises above a set threshold, an email will be sent out to alert those involved that the laser needs attention. The cooling system must also be turned off when the temperature gets too high so that dew does not form on the laser.

Implementation

An Arduino is used to control the temperature of the cooler and take temperature and humidity readings. This Arduino is controlled by a Python computer interface. When the program is launched, the user can easily connect to the Arduino and the past 2 hours of temperature and humidity data is automatically displayed in a set of graphs. The Arduino that is used is the 5V Arduino Pro Mini, shown in Figure 1. It is connected to a PTCxK-CH chassis-mount temperature controller, seen in Figure 2. The controller takes in an analog voltage as a temperature set point. This voltage is produced by a DAC (MCP4725) connected to the Arduino. The controller contains a current source connected to a thermistor, and can measure the voltage drop over that thermistor to determine its resistance. The controller has an internal control loop that heats and cools the laser with a thermoelectric cooler in order to match the voltage input to the voltage drop over the thermistor. A sample thermistor resistance vs temperature graph can be seen in Figure 3. When the Arduino receives a temperature set point from the Python interface, it will calculate the necessary resistance that the thermistor will have at that temperature, and send a corresponding voltage to the temperature controller.

The humidity detection system is built into the cooler and simply outputs a voltage corresponding to a humidity. The voltage/humidity curve is known and the Arduino can find the humidity based on that curve, and report it to the Python interface. The python interface then determines if the humidity is higher than the set threshold, and if it is, it will automatically send out emails to every specified email address. If necessary, it will also turn off the cooling system.

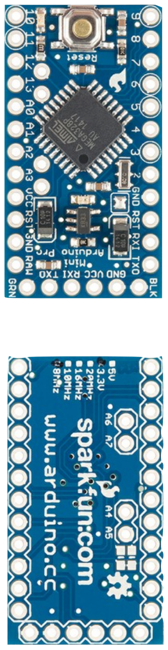


Figure 1: The Arduino Pro Mini

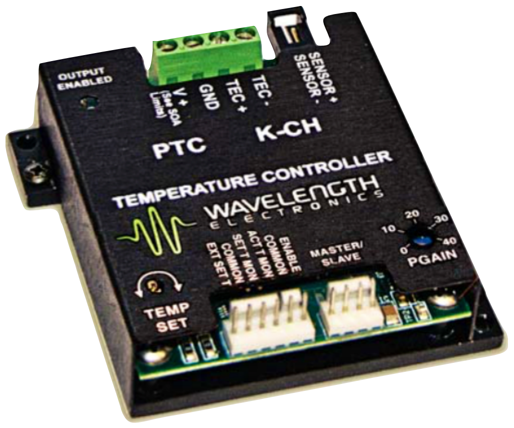


Figure 2: The PTCxK-CH

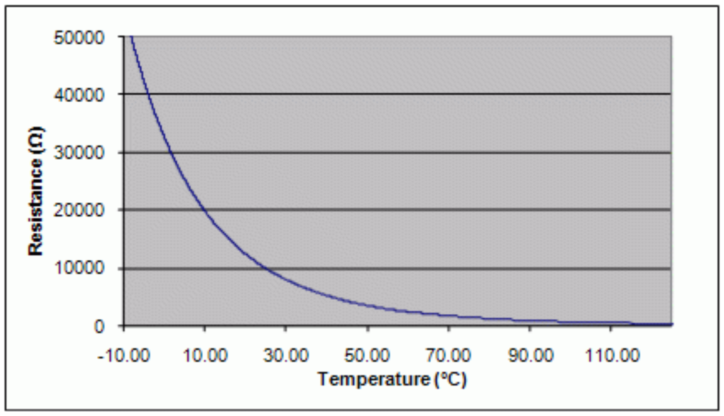


Figure 3: A sample thermistor temperature/resistance curve

Usage

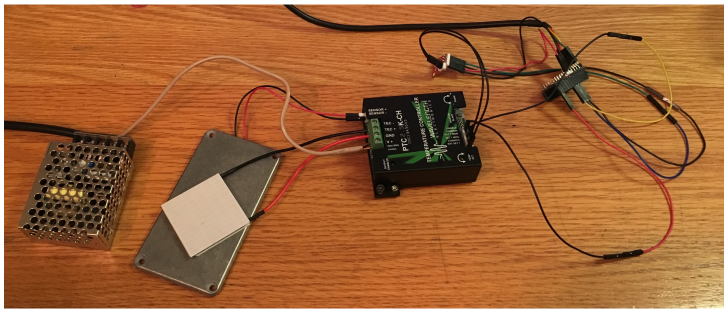
Before the controller can be used, several settings must be changed beneath the casing. First, the voltage setpoint jumper must be set to EVS, in order to allow an external voltage to set the temperature. The enable jumper must be set to EEN so the current can be turned on and off externally. The current jumper must be set to 10 μA of current over the thermistor, and the sensor type must be set to "OTHER." Finally, the current limiter dial should be set to 10, the maximum value, unless the specific TEC requires a lower current. The one I used was most efficient at 14 amps, so there was no need to limit the 2.5 amp current. See the table below for the current limit value.

POT POS.	CURRENT LIMIT VALUE IN AMPS		
	PTC2.5K-CH	PTC5K-CH	PTC10K-CH
1	In this range, the response is non-linear.		
2			
3			
4	1	2	4
5	1.25	2.5	5
6	1.5	3	6
7	1.75	3.5	7
8	2	4	8
9	2.25	4.5	9
10	2.5	5	10

To use the system, several pins on the different devices must be connected. First, the controller must be connected to a DC power source of between 5 and 30 volts. The device can produce up to 2.5 amps at maximum power. Then, the thermoelectric cooler must be connected to the TEC pins and a 10KΩ thermistor must be attached to the sensor pins. 10KΩ was chosen because a drop over a 10KΩ thermistor with 10 μA of current will be between 0 and 5 Volts for the temperatures we are measuring. This is within the analog input voltage range for the Arduino, so no external ADC is needed.

The Arduino must then be connected to the controller to ensure the temperature is set correctly. Pins A0 and A1 will be used to measure the humidity and temperature, respectively. Pin A0 will take a voltage directly off the HIH-5031 humidity sensor and convert it to a relative humidity for the GUI to display. Pin A1 should be connected to "ACT T MON" on the controller in order to read the voltage drop over the thermistor. This voltage will be converted to a temperature by the Arduino to be displayed in the GUI. In addition, the "ENABLE" pin on the controller must be connected to pin 10 on the Arduino to turn on and off the TEC current output.

The DAC must then be connected to the Arduino and the controller. Pins A2 and A3 on the Arduino should be connected to ground and VCC, respectively, on the DAC. Pins A4 and A5 on the Arduino Pro Mini are not on the edge, but rather in the middle of the device. These are the I^{2}C communications pins that can communicate with the DAC. A4 and A5 should be connected to SDA and SCL, respectively. Finally, the two output pins on the DAC - GND and OUT - should be connected to "COMMON" (either pin) and "EXT SET T" on the controller. This should complete the wiring and setup of the controller. The final setup can be seen below. The thermistor is attached to the underside of the metal plate in the image, which was used in place of the laser. The humidity sensor is not connected. In addition, the Arduino is connected to the computer via USB.

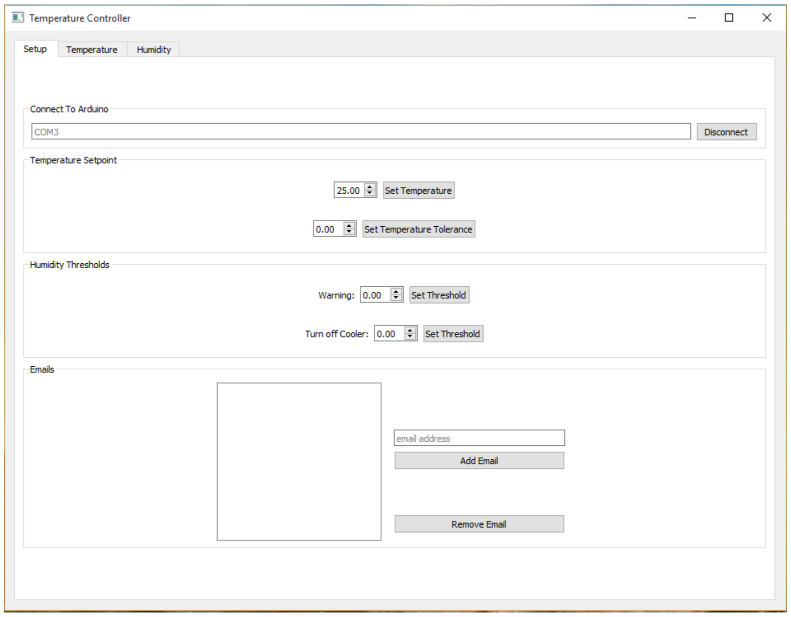


Testing

Testing the sensor produced very promising results. After the system was fully connected, the Arduino was connected to the computer and the power supply was turned on. The TEC immediately got hot on one side and cold on the other, cooling the metal plate and lowering the temperature read by the thermistor and displayed on the graph. One important thing to note was that the TEC cannot effectively cool itself, so unless it was connected to some other heat dissipation device, both sides ended up getting very hot. My testing was done exclusively with a metal pan full of ice to dissipate heat from the hot side of the device, but a large enough heat sink should produce similar results.

Setting up the GUI and connecting to the Arduino is simple. On Linux, the dependencies are listed in the README.md file in the GitHub repository. On Windows, WinPython 3.4.3.6Qt5 includes all necessary dependencies. To connect to the device on Linux, find the filename of the serial device, usually located at `/dev/USBtty0` or `/dev/USBtty1`. On Windows, the device is usually connected to COM[X], where x is some integer. This number can be determined in the Windows device manager.

To connect to the Arduino, type in the location of the device found above and click "Connect," as shown in the image below. The measured temperature and humidity data should automatically fill the graphs in the Temperature and Humidity tabs. Adding email addresses to the list will allow the GUI to send out an email when the temperature leaves the temperature tolerance range or the humidity is above the humidity threshold.



Connect To Arduino: COM3

Temperature Setpoint: 25.00

Temperature Tolerance: 0.00

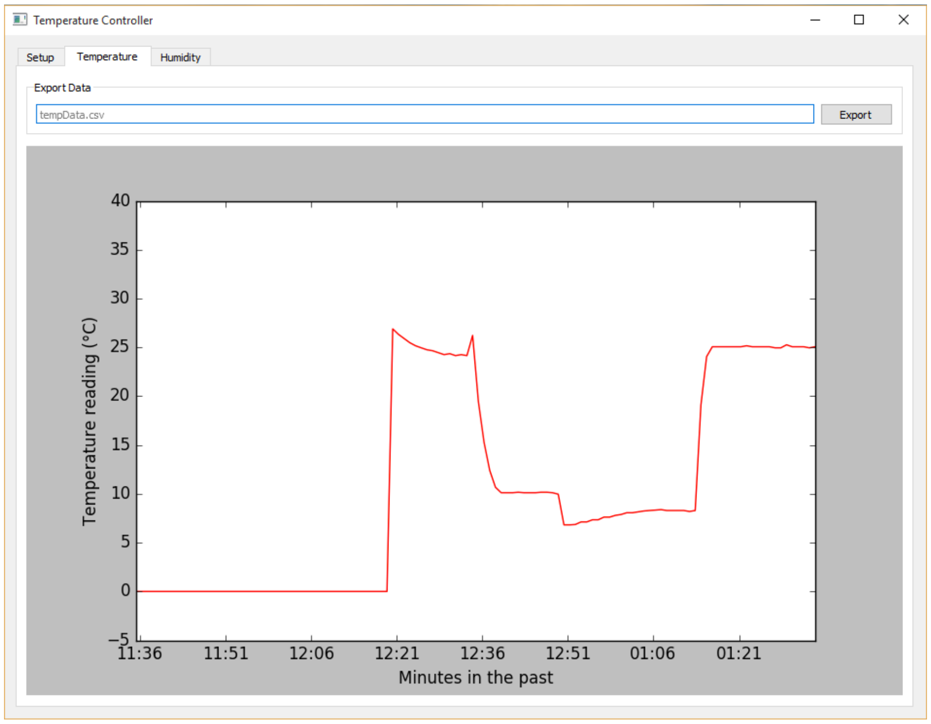
Humidity Thresholds

Warning: 0.00

Turn off Cooler: 0.00

Emails

An image of the temperature sensor output is shown below. The first “segment” of the graph was a time when the thermistor was reading data, but the TEC was not active. The thermistor was settling to room temperature, which is why the graph is slightly curved downward. Next, a steep drop is seen in the graph. This is when I set the temperature to 10°C. As you can see, the temperature reached that point relatively quickly - within 5 minutes. The next drop corresponds to when I set the temperature to 4°C. As shown in the image, his temperature was never reached. I believe the reason for this is that the controller only supports a maximum of 2.5 amps, but the TEC is most efficient at 14 amps. A smaller TEC or a higher-amperage controller should fix this issue. I found that with the current setup, 8°C is the lowest temperature I could reliably set the temperature to. Finally, the graph increases dramatically toward the end. This was when I set the temperature to 25°C. This temperature was reached in about 3 minutes and remained steady, even with a bucket of ice on top of the TEC.



Future Steps

The current state of the temperature control system is nearly finished. I would recommend to whomever takes over this project that they purchase a more powerful controller if they are planning to set temperatures lower than 8°C. The final step is testing this system with a laser attached. Based on the data that I collected, a laser should be able to be cooled very effectively with this setup.

The code for the Arduino and GUI is available on GitHub at the following URL:

https://github.com/camerongivler/TempController

Feel free to email me with any questions or feature requests at cdg26@duke.edu.