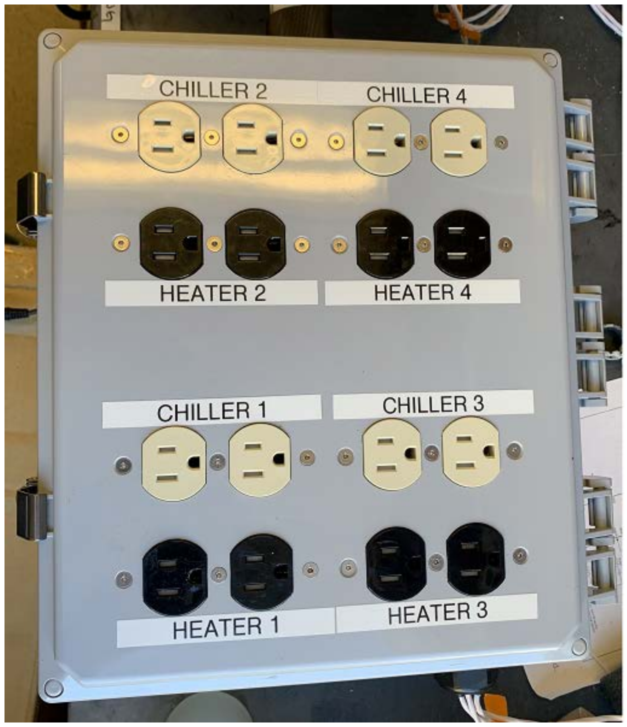
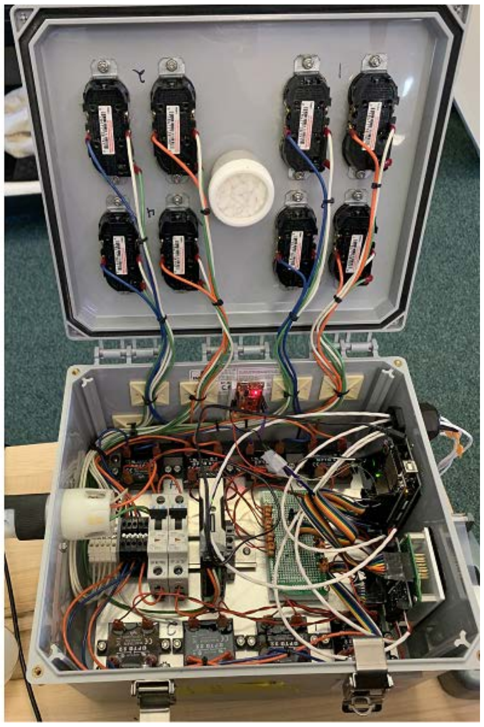
4 Port PID Controller

Proportional–Integral–Derivative





##

2021-10-14 Rev 0.7

Tom Rolander, MSEE
Mentor, Circuit Design & Software
Miller Library, Fabrication Lab
Hopkins Marine Station, Stanford University,
120 Ocean View Blvd, Pacific Grove, CA 93950

+1 831.915.9526 | rolander@stanford.edu

Summary

This PID controller continuously calculates an error value as the difference between a desired setpoint temperature and the measured temperature and applies a correction based on proportional, integral, and derivative terms. Corrections are applied by turning on/off solid state relays controlling 120V heater/cooler receptacles

The purpose of this document is to describe the hardware and software components of the 4 Port PID Controller. This documentation includes the SD Card file formats and the cabling between the Arduino Mega2560 w/ Data Logging Shield,the two daughter boards, and the ESP8266 WiFi Access Point. Also included are instructions for initial file configuration/setup, the FAULT error codes, and the process for updating the Arduino software.

The Arduino Mega2560 is used as the Microcontroller for the system. The data logger shield uses an SD Card for configuration files and logging. There are two daughter boards connected to the Microcontroller. The first one has 4 temperature sensor amplifiers, each one is connected to a 3-wire RTD probe. The second daughter board connects to the solid state relays to turn on/off the 120V heater/cooler receptacles. An ESP8266 WiFi Access Point (Sparkfun Thing Dev) is connected to the Mega2560 with a two wire serial interface to upload and download files from the SD card to a WiFi connected laptop.

The Controller temperature readings are taken approximately every 10 seconds and input to the PID algorithm. Then depending on whether the temperature is below or above the setpoint the output of the PID algorithm is applied to the heater or cooler receptacles. The temperature readings are logged approximately every 60 seconds.

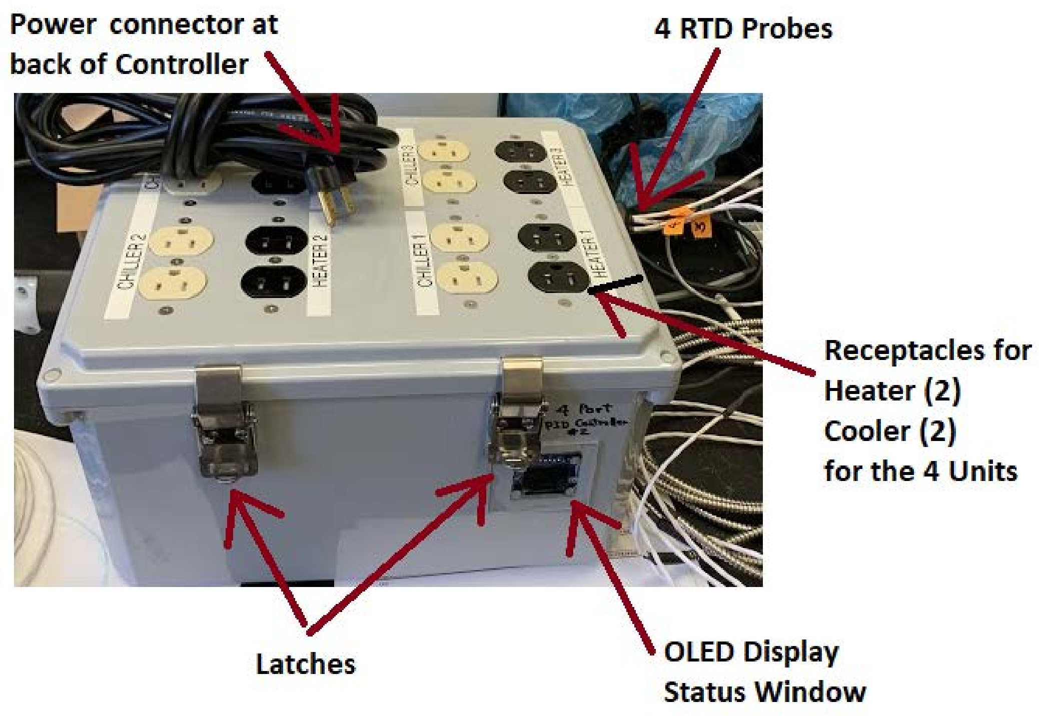
All temperature readings and settings are in degrees Centigrade.

Contents

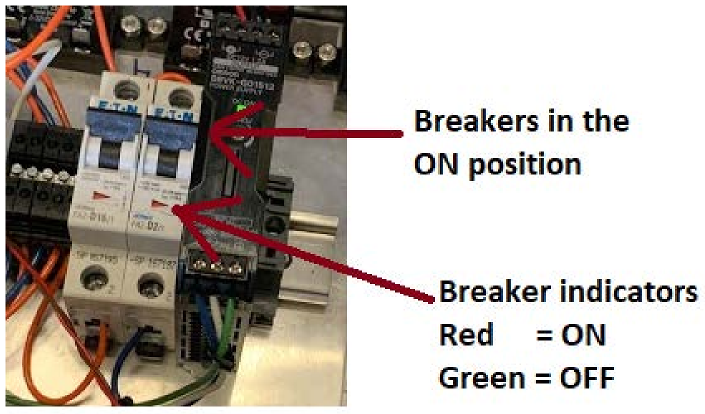
- Controller Operation
- SD Card Files
    - CLOCK.CSV
    - CONTROL.CSV
    - OFFSETS.CSV
    - SETPNT1.CSV, SETPNT2.CSV, SETPNT3.CSV, SETPNT4.CSV
    - TEST.CSV
    - LOGGING.CSV
- Controller Cabling
- Initial File Configuration and Setup
- WiFi Access Point and Python Utility Operations
  UploadFile.py
  DownloadFile.py
  ResetTomPort.py
- Working with the LOGGING.CSV File
- FAULT Error Codes
- Updating the Arduino INO file
- Formatting SD Card under Windows/Mac
- Controller Schematic
- Controller Breadboard
- ESP8266 WiFi Access Point Connections
- Controller INO Source Code
- ESP8266 WiFi Access Point INO Source Code
- Python TomPort Utilities Source Code

Controller Operation

The Controller is operated by plugging in the power connector at the back of the system and then moving the breakers into the ON position.


Power connector at back of Controller
4 RTD Probes
CHILLER 3
HEATER 3
CHILLER 2
HEATER 2
CHILLER 1
HEATER 1
4 Port PID Controller #2
Receptacles for Heater (2) Cooler (2) for the 4 Units
Latches
OLED Display Status Window

The breaker at the left controls the 120V solid state relays / heater / cooler receptacles and the right breaker controls the 12V DC to the Microcontroller. It is suggested that both breakers be turned on/off at the same time.



SD Card Files:

CLOCK.CSV (and PRVCLOCK.CSV)

The Data Logging Shield for the Arduino Mega2560 has a Real Time Clock and battery. Since there is no Internet access from the controller, the RTC must be initialized with date and time. Once the RTC date and time are initialized, the time will be maintained by the Data Logging Shield. The RTC date and time are input from an optional CLOCK.CSV file on the SD Card. If the CLOCK.CSV file is present at start up of the Controller, the PRVCLOCK.CSV file, is deleted if present, and the date and time are set by the contents of the CLOCK.CSV file. After completing this operation the CLOCK.CSV file is renamed to PRVCLOCK.CSV.

CSV File format:

```
"Year","Month","Day","Hour","Minute","Second"
2019,5,9,10,55,0
```

CONTROL.CSV

The Controller uses the ARDUINO PID software Library. There are four parameters which are used to control the PID algorithm. These values are input from the CONTROL.CSV file. It is unlikely that these parameters will need to be changed but they can be edited as required.

CSV File Format:

```
Kp,Ki,Kd,P_ON_E
2,5,1,1
```

OFFSETS.CSV

Each of the four MAX31865 RTD Temperature Sensor Amplifiers has an offset. This offset value is applied to each temperature reading. Note that the LOGGING.CSV file also includes the offset value for each reading so that the raw data can be obtained.

CSV File Format:

```
Offset1,Offset2,Offset3,Offset4
0.0,0.0,0.0,0.0
```

SETPNT1.CSV, SETPNT2.CSV, SETPNT3.CSV, SETPNT4.CSV

There are separate setpoint files for each of the four temperature probes. Each file contains 24 hours of setpoint values, 10 values per hour, one setpoint for each 6 minutes.

CSV File Format:

```
0:06,31
0:12,31
0:18,31
0:24,31
0:30,31
0:36,31
0:42,31
0:48,31
0:54,31
1:00,31
  …
  …

23:00,35
23:06,35
23:12,35
23:18,35
23:24,35
23:30,35
23:36,35
23:42,35
23:48,35
23:54,35
0:00,31
```

TEST.CSV

This CSV file allows you to test each of the Heater and Cooler units. If this file is present on the SD Card the controller will conduct the specified test sequence and then halt.

CSV File Format:

```
[Unit][H | C],[Seconds}

E.g.
1H,10
1C,10
2H,10
2C,10
3H,10
3C,10
4H,10
4C,10
```

LOGGING.CSV

This is the CSV file where the temperatures are recorded for each probe, once every minute. Each time the Controller is powered ON, a log entry of "Start up" is made. All of the parameters associated with a temperature reading are recorded in the log file.

CSV File Format:

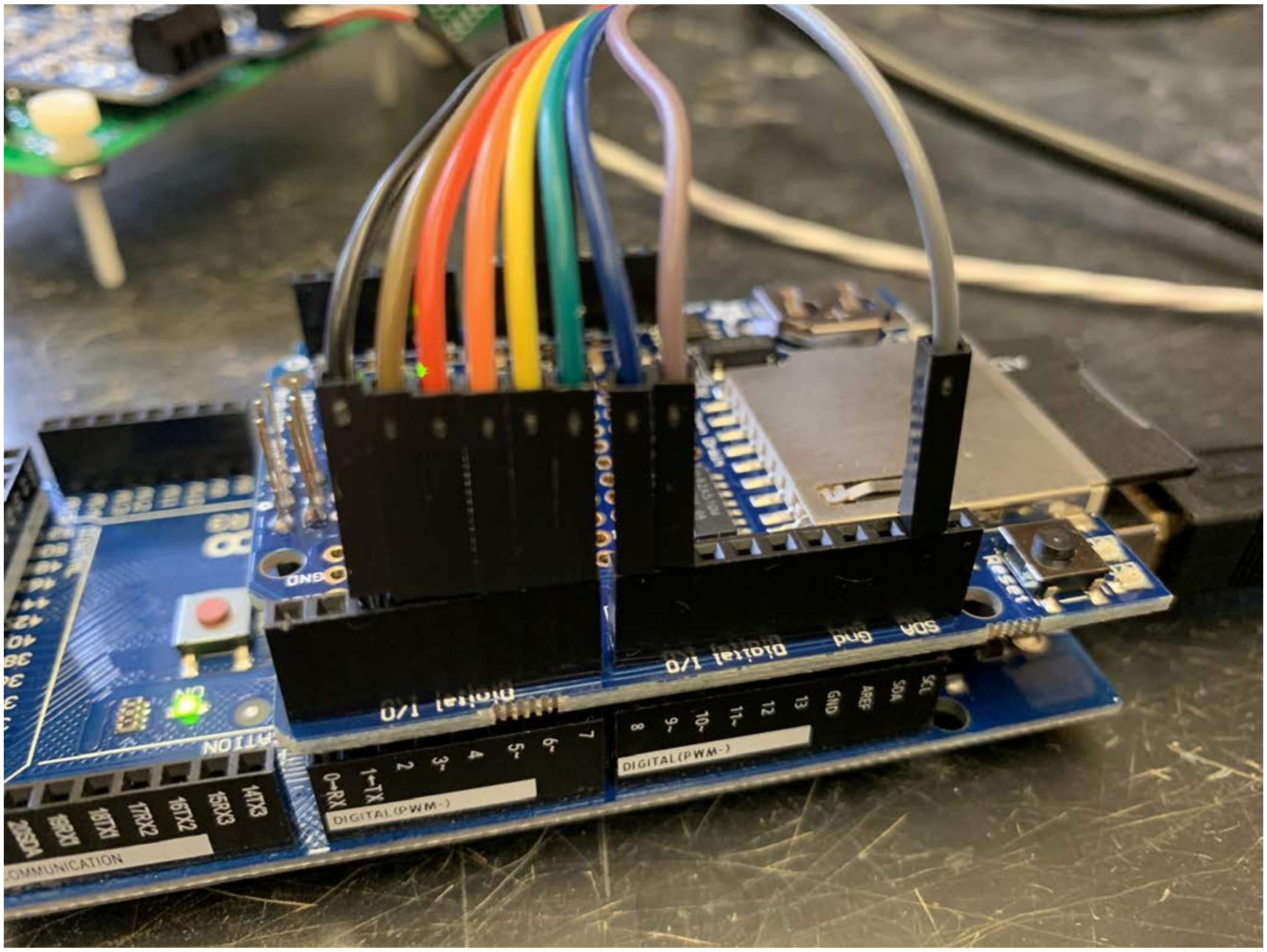
```
Date,Time\n,Unit,Setpt,Temp1,Delta,Offset,Output,DutyCycle,Dir
2019/6/27,11:54:03,Start Up,0.00,0.00,0.00,0.00,0.00,0.00%,
2019/6/27,11:54:04,Unit1,31,23.33,-0.3,23.0,-7.67,0,59.57,23.36%,Heat
2019/6/27,11:54:04,Unit2,31,31.01,0.01,0.06,0,0.00%,OFF
2019/6/27,11:54:04,Unit3,31,30.97,-0.03,-0.01,25.47,9.99%,Heat
2019/6/27,11:54:04,Unit4,31,30.97,-0.03,0.12,25.59,10.04%,Heat
…
```

Cabling between the Arduino Mega2560 and the two daughter boards

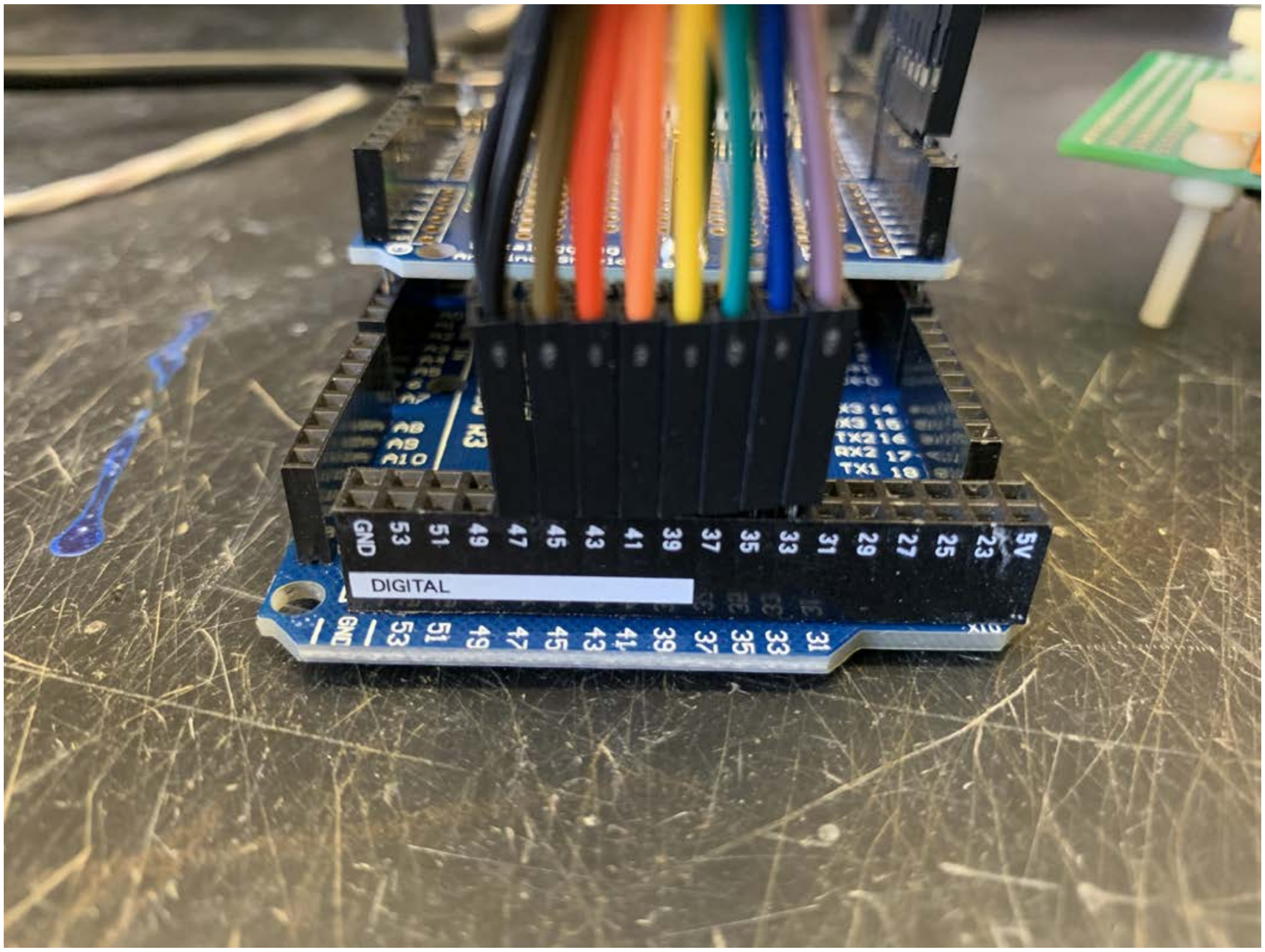
Arduino Mega2560 and Data Logging Shield (SD Card and RTC)

Arguino Mega 2560

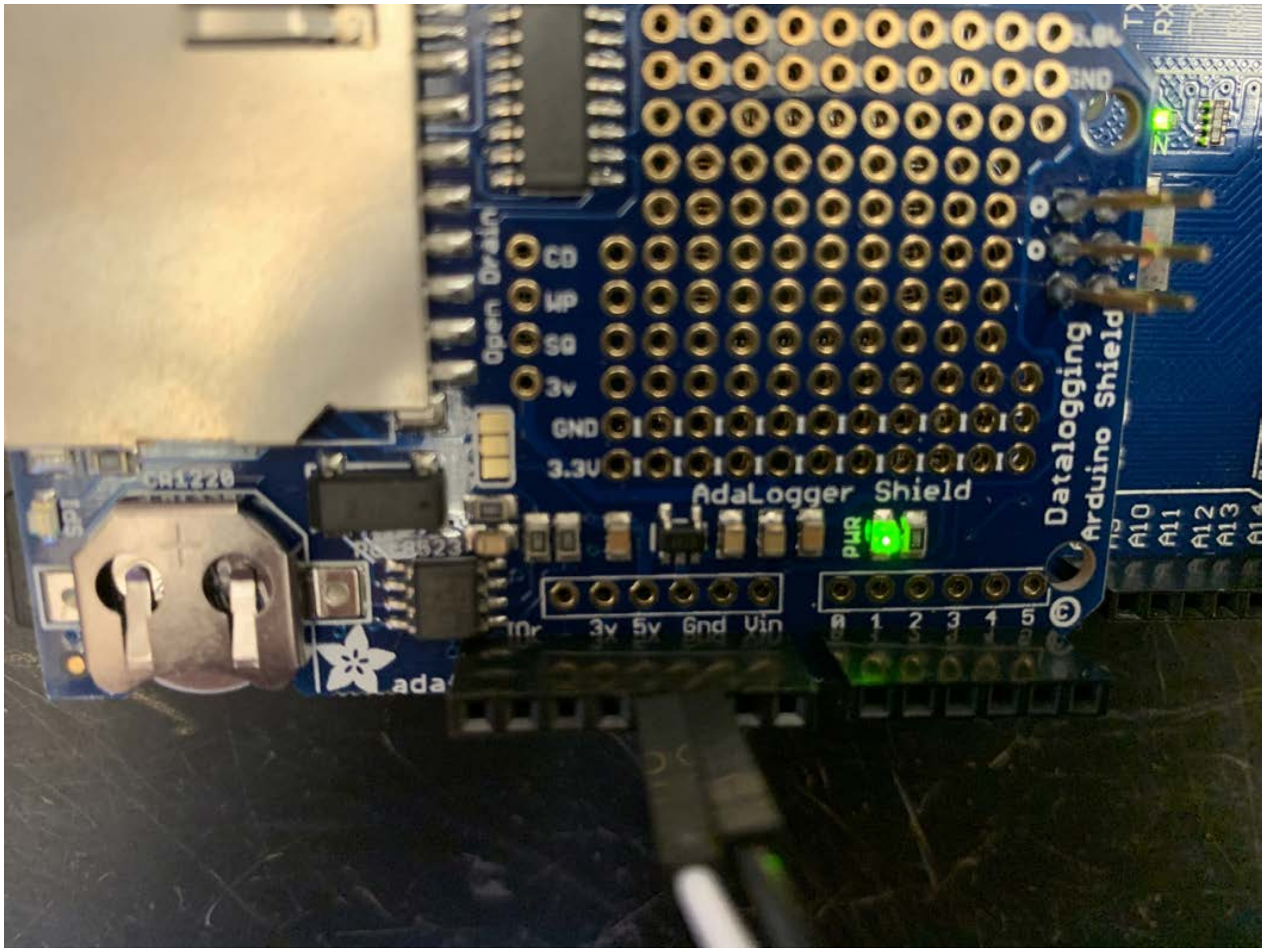
- To daughter board for solid state relays
  - Digital Pins 2-9 (8 outputs for the 4 Heaters + 4 Coolers)
    - Pin 2 Heater 1
    - Pin 3 Cooler 1
    - Pin 4 Heater 2
    - Pin 5 Cooler 2
    - Pin 6 Heater 3
    - Pin 7 Cooler 3
    - Pin 8 Heater 4
    - Pin 9 Cooler 4
  - Gnd



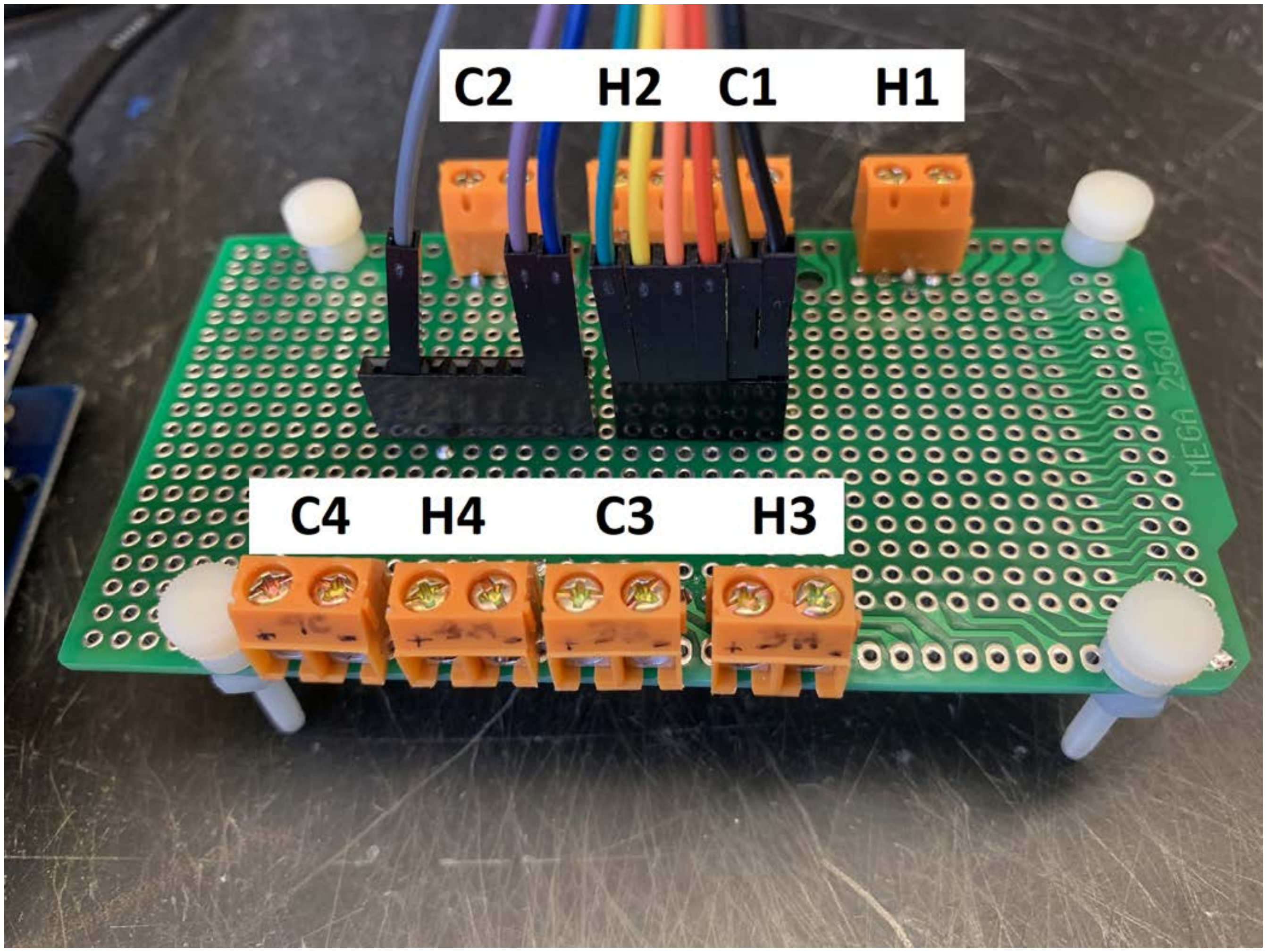
To Max31865 RTD daughter board
Digital Pins 32-47



To Max31865 RTD daughter board
Gnd

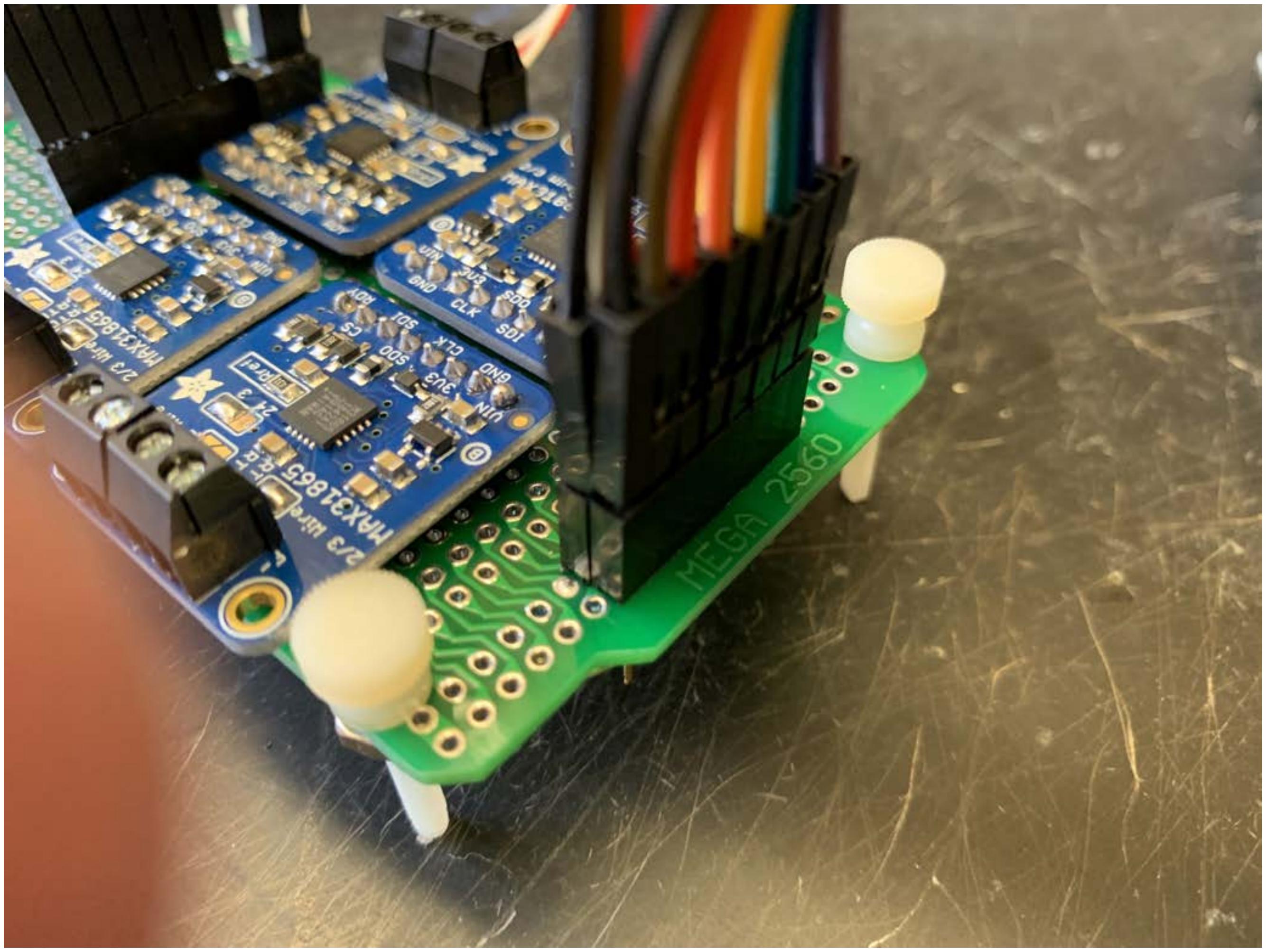


Daughter board for solid state relays

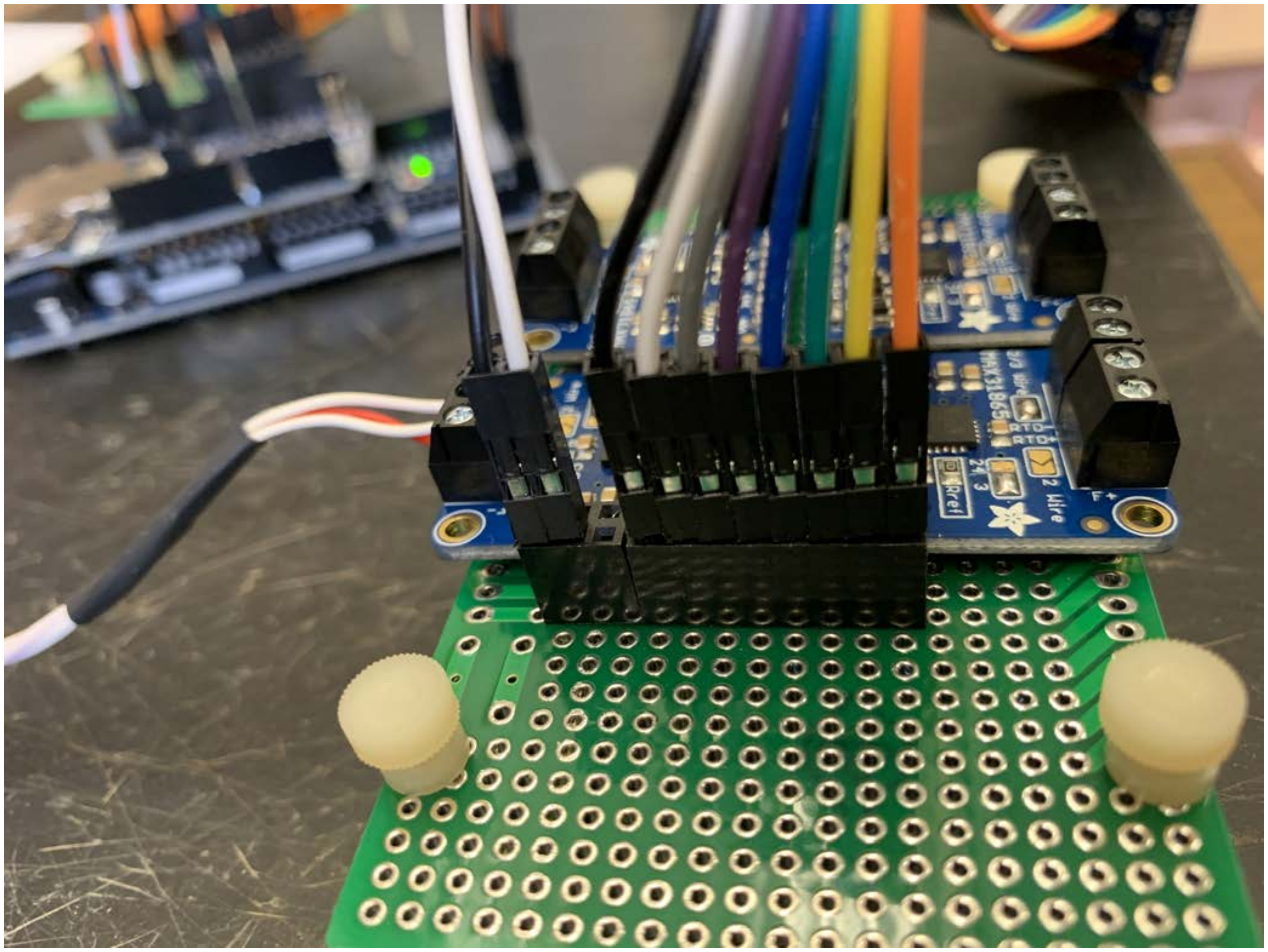


Max31865 RTD daughter board

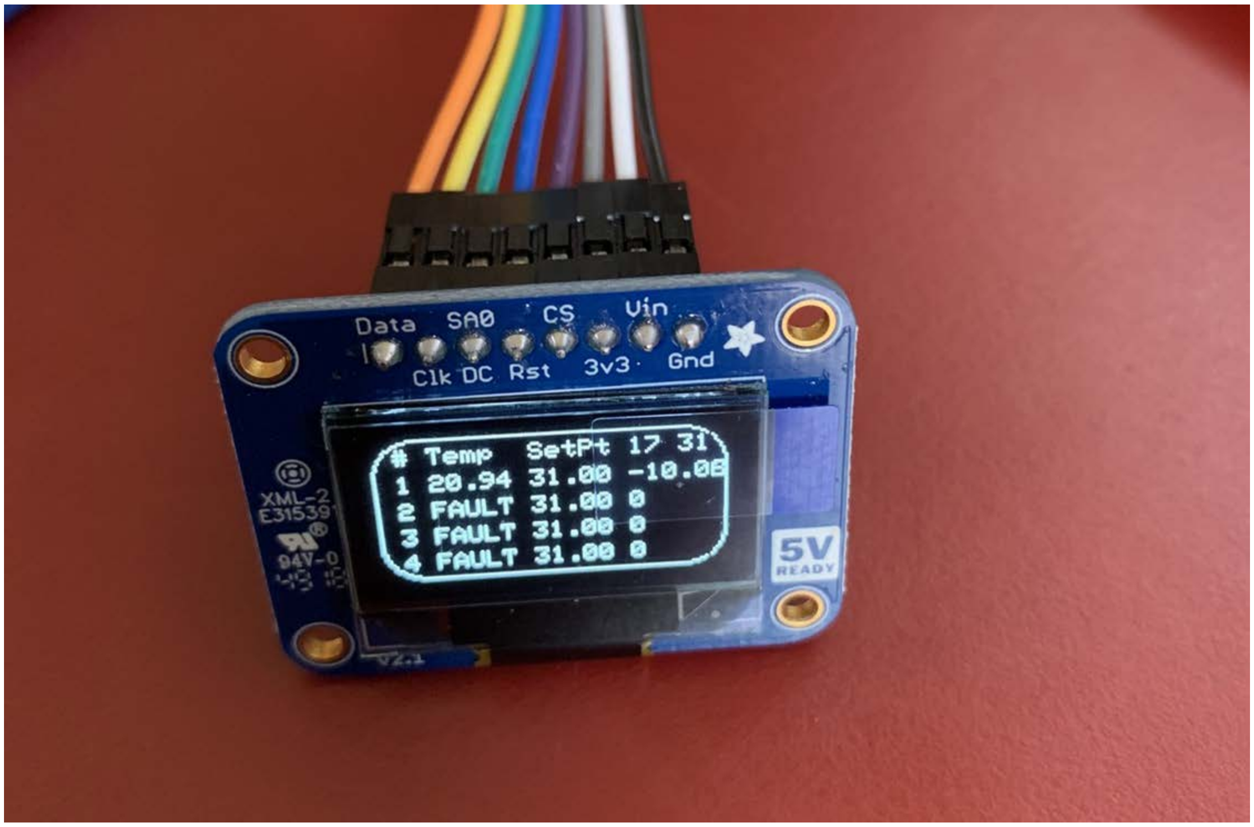
Connection from Arduino Mega2560



Connection to OLED Display (128x64)



OLED Display (128x64)



```
# Temp  SetPt 17 31
1 20.94 31.00 -10.08
2 FAULT 31.00 0
3 FAULT 31.00 0
4 FAULT 31.00 0
```

Initial File Configuration and Setup

CLOCK.CSV

The Controller does not have a connection to the Internet to obtain current date and time. However, a real time clock (RTC) is included on the data logging shield. This RTC can be set using a CLOCK.CSV file. The contents of this file are described in the file format section CLOCK.CSV.

Before powering ON the Controller create a CLOCK.CSV file on the SD Card with the desired Date and Time. The SD Card has a PRVCLOCK.CSV file which can be used as a template for the CLOCK.CSV file. Since it will take a few minutes between the time you create, edit and store the CLOCK.CSV file on the SD Card, enter a time that is ahead by several minutes. With the SD Card inserted in the Controller, turn it ON a few seconds before the time which you have edited in the CLOCK.CSV file.

OFFSETS.CSV

This file is described in the file format for OFFSETS.CSV. The offsets for each unit can be determined by obtaining readings from the RTD and comparing them to a thermometer or other accurate temperature reading in the tank for the unit. Alternatively all four of the RTD probes could be put into the same tank and then an average temperature determined. The RTD offset would be the difference between the average and the reading from the RTD probe.

When performing this calibration procedure the SETPNTx.CSV files should all be set to the same value for the 24 hour period. The recommended value is midway between the desired low and high setpoints.

WiFi Access Point

The WiFi Access Point is used to facilitate downloading LOGGING.CSV files for a specified Date/Time and uploading CSV files, e.g. CLOCK.CSV, OFFSETS.CSV, SETPNTx.CSV. There is also a command URL to reset the TomPort and initialize the system with updated CSV files (e.g. OFFSETS.CSV, etc).

When the TomPort is powered up your laptop computer should display TOMPORT01 and/or TOMPORT02 as available networks.

TOMPORT01 has a fixed IP address of 192.168.4.1
TOMPORT02 has a fixed IP address of 192.168.4.2

The operation of the WiFi Access Point can be confirmed with either a laptop or a smartphone using the following steps:

- Connect to either TOMPORT01 (or TOMPORT02, this example is for TOMPORT01)

- Password: HMS-TOMPORT

- With a browser enter the following URL:
  http://192.168.4.1

- You will see the following display
  Invalid Request.
  Try /led/1, or /led/0.

- Entering the following:
  http://192.168.4.1/led/1

- You will see the following display
  The LED is now on
  With the TomPort unit open you will see a brightly lit LED on the ESP8266
  *** NOTE *** The power LED is a separate Red LED

- Entering the following:
  http://192.168.4.1/led/0

- You will see the following display
  The LED is now off
  With the TomPort unit open you will see the LED will be off on the ESP8266

Python Utility Applications

There are three Python programs provided to upload files, download files, and reset the TomPort. They are each described in the following pages.

All of the Python programs require that you have installed the "requests" library.

If you do not have the requests library installed the following steps will install the library on a Mac or Windows computer:

Mac

```
curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py

python3 get-pip.py

python3 -m pip install requests
```

Windows

```
python -m pip install requests
```

UploadFile.py

Command line:
python UploadFile.py

Usage: python UploadFile.py TomPort filename
Where:
TomPort is '1' or '2'
filename is one of the following:
OFFSETS.CSV
CLOCK.CSV
SETPNT1.CSV
SETPNT2.CSV
SETPNT3.CSV
SETPNT4.CSV

Sample Upload: OFFSETS.CSV

```
>python UploadFile.py 1 OFFSETS.CSV

Name of Python script: UploadFile.py

Arguments passed:
TomPort [ 1 ]
UploadFile [ OFFSETS.CSV ]
len = 70
http://192.168.4.1/?BOF
200
beg = 0
nxt = 70
len = 70
http://192.168.4.1/?f=OFFSETS.CSV&o=0&s=70&d=Offset1%2COffset2%2COffset3%2COffset4%0D%0A0.0%2C0.
0%2C0.0%2C0.0%0D%0A
200
http://192.168.4.1/?EOF
200
```

Sample Upload: SETPNT1.CSV
*** NOTE *** This operation will upload the file in 1K chunks

```
>python UploadFile.py 1 SETPNT1.CSV

Name of Python script: UploadFile.py

Arguments passed:
TomPort [ 1 ]
UploadFile [ SETPNT1.CSV ]
len = 4482
http://192.168.4.1/?BOF
200
```

```
beg = 0
nxt = 1024
len = 1024
http://192.168.4.1/?f=SETPNT1.CSV&o=0&s=1024&d=0%3A06%2C30%0D%0A0%3A12%2C30%0D%0A0%3A18%
2C30%0D%0A0%3A24%2C30%0D%0A0%3A30%2C30%0D%0A0%3A36%2C30%0D%0A0%3A42%2C30%0D%0
A0%3A48%2C30%0D%0A0%3A54%2C30%0D%0A1%3A00%2C30%0D%0A1%3A06%2C30%0D%0A1%3A12%2C
30%0D%0A1%3A18%2C30%0D%0A1%3A24%2C30%0D%0A1%3A30%2C30%0D%0A1%3A36%2C30%0D%0A1
%3A42%2C30%0D%0A1%3A48%2C30%0D%0A1%3A54%2C30%0D%0A2%3A00%2C30%0D%0A2%3A06%2C30
%0D%0A2%3A12%2C30%0D%0A2%3A18%2C30%0D%0A2%3A24%2C30%0D%0A2%3A30%2C30%0D%0A2%3
A36%2C30%0D%0A2%3A42%2C30%0D%0A2%3A48%2C30%0D%0A2%3A54%2C30%0D%0A3%3A00%2C30%0
D%0A3%3A06%2C30%0D%0A3%3A12%2C30%0D%0A3%3A18%2C30%0D%0A3%3A24%2C30%0D%0A3%3A30
%2C30%0D%0A3%3A36%2C30%0D%0A3%3A42%2C30%0D%0A3%3A48%2C30%0D%0A3%3A54%2C30%0D%
0A4%3A00%2C30%0D%0A4%3A06%2C30%0D%0A4%3A12%2C30%0D%0A4%3A18%2C30%0D%0A4%3A24%2
C30%0D%0A4%3A30%2C30%0D%0A4%3A36%2C30%0D%0A4%3A42%2C30%0D%0A4%3A48%2C30%0D%0A
4%3A54%2C30%0D%0A5%3A00%2C30%0D%0A5%3A06%2C30%0D%0A5%3A12%2C30%0D%0A5%3A18%2C3
0%0D%0A5%3A24%2C30%0D%0A5%3A30%2C30%0D%0A5%3A36%2C30%0D%0A5%3A42%2C30%0D%0A5%
3A48%2C30%0D%0A5%3A54%2C30%0D%0A6%3A00%2C30%0D%0A6%3A
200
beg = 1024
nxt = 2048
len = 1024
http://192.168.4.1/?f=SETPNT1.CSV&o=1024&s=1024&d=06%2C30%0D%0A6%3A12%2C30%0D%0A6%3A18%2
C30%0D%0A6%3A24%2C30%0D%0A6%3A30%2C30%0D%0A6%3A36%2C30%0D%0A6%3A42%2C30%0D%0A
6%3A48%2C30%0D%0A6%3A54%2C30%0D%0A7%3A00%2C30%0D%0A7%3A06%2C30%0D%0A7%3A12%2C3
0%0D%0A7%3A18%2C30%0D%0A7%3A24%2C30%0D%0A7%3A30%2C30%0D%0A7%3A36%2C30%0D%0A7%
3A42%2C30%0D%0A7%3A48%2C30%0D%0A7%3A54%2C30%0D%0A8%3A00%2C30%0D%0A8%3A06%2C30%
0D%0A8%3A12%2C30%0D%0A8%3A18%2C30%0D%0A8%3A24%2C30%0D%0A8%3A30%2C30%0D%0A8%3A3
6%2C30%0D%0A8%3A42%2C30%0D%0A8%3A48%2C30%0D%0A8%3A54%2C30%0D%0A9%3A00%2C30%0D
%0A9%3A06%2C30%0D%0A9%3A12%2C30%0D%0A9%3A18%2C30%0D%0A9%3A24%2C30%0D%0A9%3A30
%2C30%0D%0A9%3A36%2C30%0D%0A9%3A42%2C30%0D%0A9%3A48%2C30%0D%0A9%3A54%2C30%0D%
0A10%3A00%2C30.12903226%0D%0A10%3A06%2C30.25806452%0D%0A10%3A12%2C30.38709677%0D%0A1
0%3A18%2C30.51612903%0D%0A10%3A24%2C30.64516129%0D%0A10%3A30%2C30.77419355%0D%0A10%3
A36%2C30.90322581%0D%0A10%3A42%2C31.03225806%0D%0A10%3A48%2C31.16129032%0D%0A10%3A54
%2C31.29032258%0D%0A11%3A00%2C31.41935484%0D%0A11%3A06%2C31.5483871%0D%0A11%3A12%2C3
1.67741935%0D%0A11%3A18%2C31.80
200
beg = 2048
nxt = 3072
len = 1024
http://192.168.4.1/?f=SETPNT1.CSV&o=2048&s=1024&d=645161%0D%0A11%3A24%2C31.93548387%0D%0A11
%3A30%2C32.06451613%0D%0A11%3A36%2C32.19354839%0D%0A11%3A42%2C32.32258065%0D%0A11%3A
48%2C32.4516129%0D%0A11%3A54%2C32.58064516%0D%0A12%3A00%2C32.70967742%0D%0A12%3A06%2
C32.83870968%0D%0A12%3A12%2C32.96774194%0D%0A12%3A18%2C33.09677419%0D%0A12%3A24%2C33.
22580645%0D%0A12%3A30%2C33.35483871%0D%0A12%3A36%2C33.48387097%0D%0A12%3A42%2C33.612
90323%0D%0A12%3A48%2C33.74193548%0D%0A12%3A54%2C33.87096774%0D%0A13%3A00%2C34%0D%0
A13%3A06%2C34%0D%0A13%3A12%2C34%0D%0A13%3A18%2C34%0D%0A13%3A24%2C34%0D%0A13%3A
30%2C34%0D%0A13%3A36%2C34%0D%0A13%3A42%2C34%0D%0A13%3A48%2C34%0D%0A13%3A54%2C34
%0D%0A14%3A00%2C34%0D%0A14%3A06%2C34%0D%0A14%3A12%2C34%0D%0A14%3A18%2C34%0D%0A
14%3A24%2C34%0D%0A14%3A30%2C34%0D%0A14%3A36%2C34%0D%0A14%3A42%2C34%0D%0A14%3A48
%2C34%0D%0A14%3A54%2C34%0D%0A15%3A00%2C34%0D%0A15%3A06%2C34%0D%0A15%3A12%2C34%
0D%0A15%3A18%2C34%0D%0A15%3A24%2C34%0D%0A15%3A30%2C34%0D%0A15%3A36%2C34%0D%0A1
5%3A42%2C34%0D%0A15%3A48%2C34%0D%0A15%3A54%2C34%0D%0A16%3A00%2C30%0D%0A16%3A06
%2C30%0D%0A16%3A
200
beg = 3072
nxt = 4096
```

```
len = 1023
http://192.168.4.1/?f=SETPNT1.CSV&o=3072&s=1023&d=12%2C30%0D%0A16%3A18%2C30%0D%0A16%3A24%
2C30%0D%0A16%3A30%2C30%0D%0A16%3A36%2C30%0D%0A16%3A42%2C30%0D%0A16%3A48%2C30%0
D%0A16%3A54%2C30%0D%0A17%3A00%2C30%0D%0A17%3A06%2C30%0D%0A17%3A12%2C30%0D%0A17
%3A18%2C30%0D%0A17%3A24%2C30%0D%0A17%3A30%2C30%0D%0A17%3A36%2C30%0D%0A17%3A42%
2C30%0D%0A17%3A48%2C30%0D%0A17%3A54%2C30%0D%0A18%3A00%2C30%0D%0A18%3A06%2C30%0
D%0A18%3A12%2C30%0D%0A18%3A18%2C30%0D%0A18%3A24%2C30%0D%0A18%3A30%2C30%0D%0A18
%3A36%2C30%0D%0A18%3A42%2C30%0D%0A18%3A48%2C30%0D%0A18%3A54%2C30%0D%0A19%3A00%
2C30%0D%0A19%3A06%2C30%0D%0A19%3A12%2C30%0D%0A19%3A18%2C30%0D%0A19%3A24%2C30%0
D%0A19%3A30%2C30%0D%0A19%3A36%2C30%0D%0A19%3A42%2C30%0D%0A19%3A48%2C30%0D%0A19
%3A54%2C30%0D%0A20%3A00%2C30%0D%0A20%3A06%2C30%0D%0A20%3A12%2C30%0D%0A20%3A18%
2C30%0D%0A20%3A24%2C30%0D%0A20%3A30%2C30%0D%0A20%3A36%2C30%0D%0A20%3A42%2C30%0
D%0A20%3A48%2C30%0D%0A20%3A54%2C30%0D%0A21%3A00%2C30%0D%0A21%3A06%2C30%0D%0A21
%3A12%2C30%0D%0A21%3A18%2C30%0D%0A21%3A24%2C30%0D%0A21%3A30%2C30%0D%0A21%3A36%
2C30%0D%0A21%3A42%2C30%0D%0A21%3A48%2C30%0D%0A21
200
beg = 4095
nxt = 4482
len = 387
http://192.168.4.1/?f=SETPNT1.CSV&o=4095&s=387&d=%3A54%2C30%0D%0A22%3A00%2C30%0D%0A22%3A
06%2C30%0D%0A22%3A12%2C30%0D%0A22%3A18%2C30%0D%0A22%3A24%2C30%0D%0A22%3A30%2C30
%0D%0A22%3A36%2C30%0D%0A22%3A42%2C30%0D%0A22%3A48%2C30%0D%0A22%3A54%2C30%0D%0A
23%3A00%2C30%0D%0A23%3A06%2C30%0D%0A23%3A12%2C30%0D%0A23%3A18%2C30%0D%0A23%3A24
%2C30%0D%0A23%3A30%2C30%0D%0A23%3A36%2C30%0D%0A23%3A42%2C30%0D%0A23%3A48%2C30%
0D%0A23%3A54%2C30%0D%0A0%3A00%2C30
200
http://192.168.4.1/?EOF
200
```

DownloadFile.py

Command line:
python DownloadFile.py

Usage: python DownloadFile.py TomPort filename xx/xx/xx,xx:
Where:

TomPort is '1' or '2'
filename is name for the downloaded CSV logging file
xx/xx/xx,xx: is the yr/mo/dy,hr:
*** NOTE *** all of the "xx" fields must NOT have a leading zero

Sample Download:

```
>python DownloadFile.py 1 TP1_2021-10-13_15.CSV 2021/10/13,15:

Arguments passed:
TomPort [ 1 ]
DownloadFile [ TP1_2021-10-13_15.CSV ]
xx/xx/xx,xx: [ 2021/10/13,15: ]
http://192.168.4.1/?h=2021/10/13%2C15%3A
200
2021/10/13,15:0:34,Unit1,34.00,34.27,0.27,0.00,255.00,100.00%,COOL (completed)
2021/10/13,15:0:34,Unit2,34.00,34.24,0.24,0.00,255.00,100.00%,COOL (waiting)
2021/10/13,15:0:34,Unit3,34.00,34.03,0.03,0.00,0.00,0.00%,OFF
2021/10/13,15:0:34,Unit4,34.00,34.00,-0.00,0.00,255.00,100.00%,COOL (waiting)
2021/10/13,15:1:39,Unit1,34.00,34.20,0.20,0.00,255.00,100.00%,COOL (completed)
2021/10/13,15:1:39,Unit2,34.00,34.17,0.17,0.00,255.00,100.00%,COOL (waiting)
2021/10/13,15:1:39,Unit3,34.00,34.03,0.03,0.00,255.00,100.00%,COOL (waiting)
2021/10/13,15:1:39,Unit4,34.00,34.00,-0.00,0.00,255.00,100.00%,COOL (waiting)
…
…
```

ResetTomPort.py

Command line:
python ResetTomPort.py

Usage: python ResetTomPort.py TomPort
Where:
    TomPort is '1' or '2'

Sample ResetTomPort:

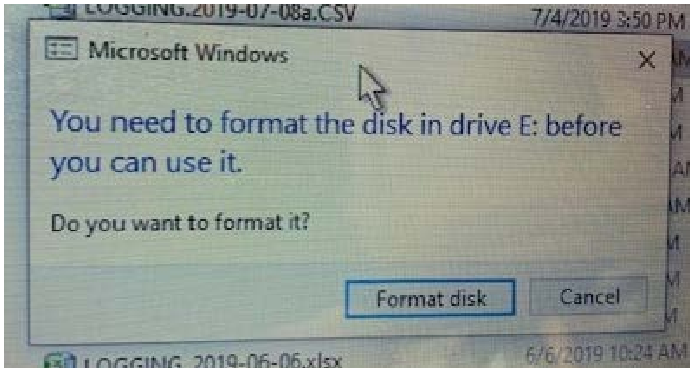
```
>python ResetTomPort.py 1

Arguments passed:
TomPort [ 1 ]
http://192.168.4.1/?RESET
200
Line sent
```

Working with the LOGGING.CSV File

The LOGGING.CSV file contains a log of temperature readings taken approximately each 60 seconds for each of the 4 RTD probes. An entry is also made in the log file each time the Controller is powered ON.

The SD Card can be removed from the Controller and inserted in a laptop computer. Copy the LOGGING.CSV file to the laptop. It may be desirable to edit the LOGGING.CSV file on the SD Card to trim the data. If the data is trimmed, leave the first line in the CSV file containing the column headings.

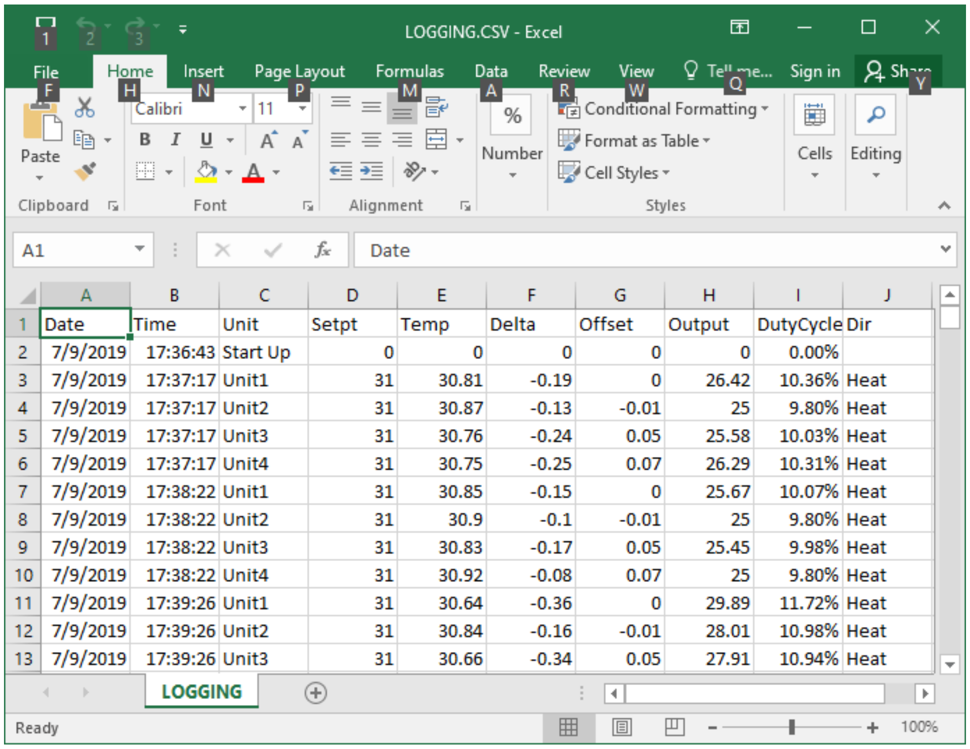
When the SD Card is inserted in a Windows / Mac computer you may see a warning to format the disk. Do not format the disk, select "Cancel", remove the SD Card and reinsert it in your laptop.



You need to format the disk in drive E: before you can use it.

Do you want to format it?

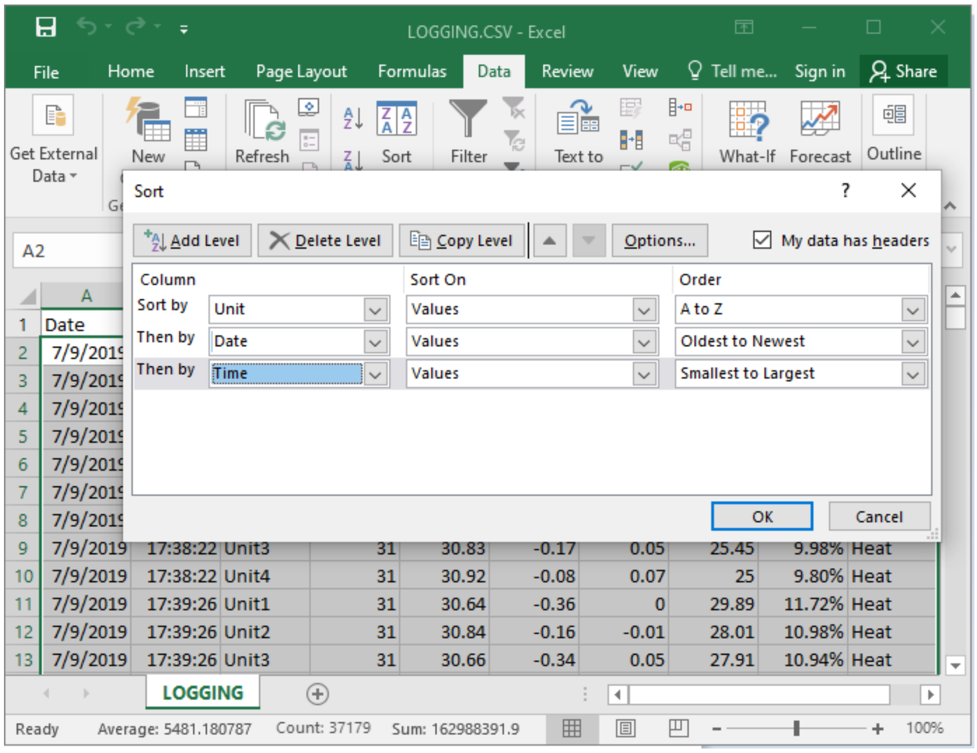
Sample of LOGGING.CSV



Date	Time	Unit	Setpt	Temp	Delta	Offset	Output	DutyCycle	Dir
7/9/2019	17:36:43	Start Up	0	0	0	0	0	0.00%	
7/9/2019	17:37:17	Unit1	31	30.81	-0.19	0	26.42	10.36%	Heat
7/9/2019	17:37:17	Unit2	31	30.87	-0.13	-0.01	25	9.80%	Heat
7/9/2019	17:37:17	Unit3	31	30.76	-0.24	0.05	25.58	10.03%	Heat
7/9/2019	17:37:17	Unit4	31	30.75	-0.25	0.07	26.29	10.31%	Heat
7/9/2019	17:38:22	Unit1	31	30.85	-0.15	0	25.67	10.07%	Heat
7/9/2019	17:38:22	Unit2	31	30.9	-0.1	-0.01	25	9.80%	Heat
7/9/2019	17:38:22	Unit3	31	30.83	-0.17	0.05	25.45	9.98%	Heat
7/9/2019	17:38:22	Unit4	31	30.92	-0.08	0.07	25	9.80%	Heat
7/9/2019	17:39:26	Unit1	31	30.64	-0.36	0	29.89	11.72%	Heat
7/9/2019	17:39:26	Unit2	31	30.84	-0.16	-0.01	28.01	10.98%	Heat
7/9/2019	17:39:26	Unit3	31	30.66	-0.34	0.05	27.91	10.94%	Heat

Note that the temperature readings are sequentially recorded for each of the 4 RTD probes.

The next step in processing the temperature readings is to sort the file into the individual Units by date and time.



Sort

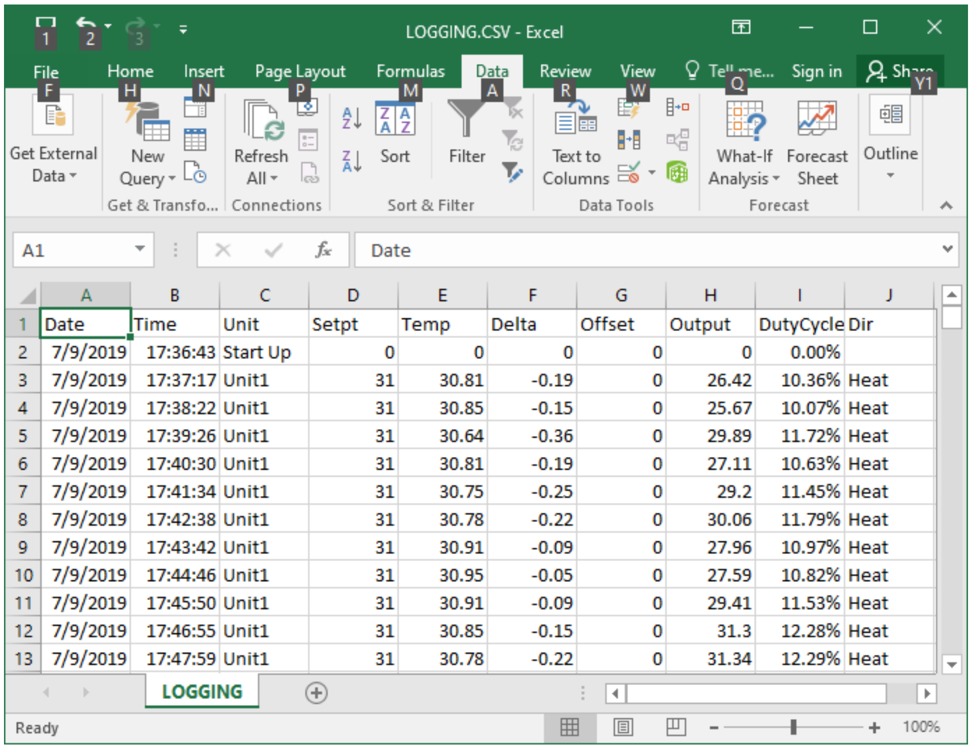
Add Level | Delete Level | Copy Level | Options... | ☑ My data has headers

Column		Sort On	Order
Sort by	Unit	Values	A to Z
Then by	Date	Values	Oldest to Newest
Then by	Time	Values	Smallest to Largest

OK | Cancel

	A	B	C	D	E	F	G	H	I	J
1	Date									
2	7/9/2019									
3	7/9/2019									
4	7/9/2019									
5	7/9/2019									
6	7/9/2019									
7	7/9/2019									
8	7/9/2019									
9	7/9/2019	17:38:22	Unit3	31	30.83	-0.17	0.05	25.45	9.98%	Heat
10	7/9/2019	17:38:22	Unit4	31	30.92	-0.08	0.07	25	9.80%	Heat
11	7/9/2019	17:39:26	Unit1	31	30.64	-0.36	0	29.89	11.72%	Heat
12	7/9/2019	17:39:26	Unit2	31	30.84	-0.16	-0.01	28.01	10.98%	Heat
13	7/9/2019	17:39:26	Unit3	31	30.66	-0.34	0.05	27.91	10.94%	Heat

All of the "Start up" times will appear first, followed by Unit1, etc.

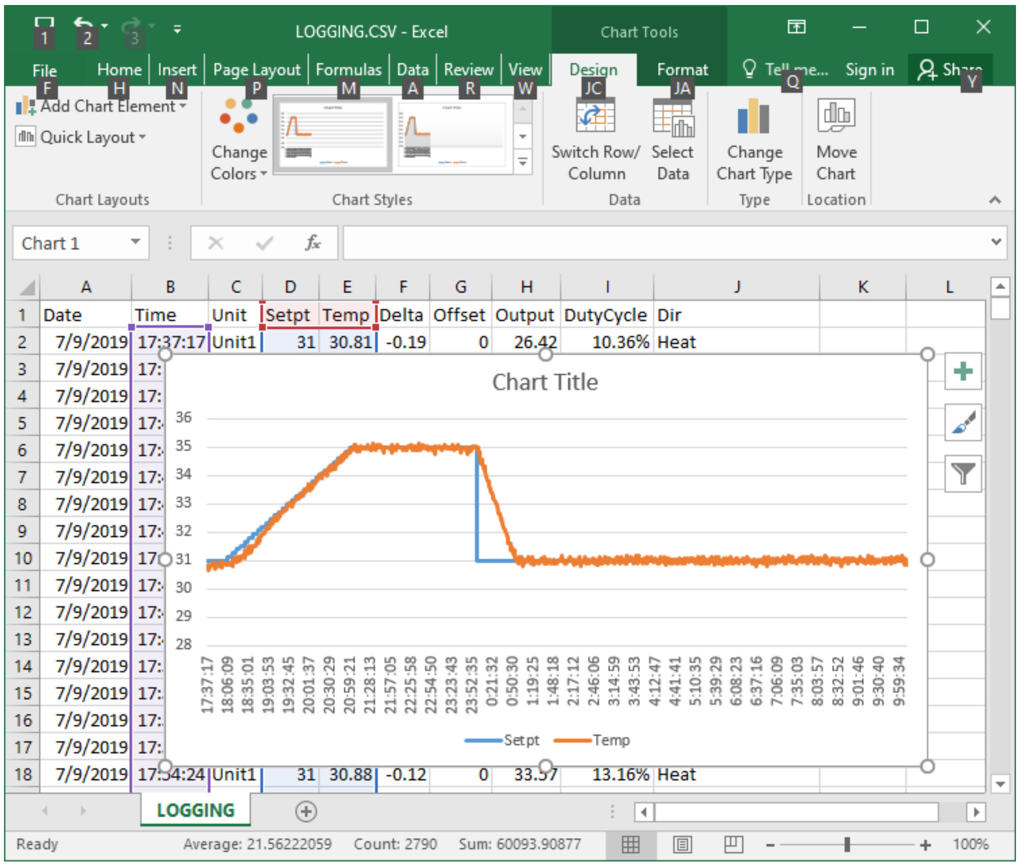


A1 | Date

	A	B	C	D	E	F	G	H	I	J
1	Date	Time	Unit	Setpt	Temp	Delta	Offset	Output	DutyCycle	Dir
2	7/9/2019	17:36:43	Start Up	0	0	0	0	0	0.00%	
3	7/9/2019	17:37:17	Unit1	31	30.81	-0.19	0	26.42	10.36%	Heat
4	7/9/2019	17:38:22	Unit1	31	30.85	-0.15	0	25.67	10.07%	Heat
5	7/9/2019	17:39:26	Unit1	31	30.64	-0.36	0	29.89	11.72%	Heat
6	7/9/2019	17:40:30	Unit1	31	30.81	-0.19	0	27.11	10.63%	Heat
7	7/9/2019	17:41:34	Unit1	31	30.75	-0.25	0	29.2	11.45%	Heat
8	7/9/2019	17:42:38	Unit1	31	30.78	-0.22	0	30.06	11.79%	Heat
9	7/9/2019	17:43:42	Unit1	31	30.91	-0.09	0	27.96	10.97%	Heat
10	7/9/2019	17:44:46	Unit1	31	30.95	-0.05	0	27.59	10.82%	Heat
11	7/9/2019	17:45:50	Unit1	31	30.91	-0.09	0	29.41	11.53%	Heat
12	7/9/2019	17:46:55	Unit1	31	30.85	-0.15	0	31.3	12.28%	Heat
13	7/9/2019	17:47:59	Unit1	31	30.78	-0.22	0	31.34	12.29%	Heat

To analyze the data, copy the data for each Unit into a separate XLS file.

Here is an example of plotting the data for Unit1.



	A	B	C	D	E	F	G	H	I	J
1	Date	Time	Unit	Setpt	Temp	Delta	Offset	Output	DutyCycle	Dir
2	7/9/2019	17:37:17	Unit1	31	30.81	-0.19	0	26.42	10.36%	Heat
3	7/9/2019	17:								
4	7/9/2019	17:								
5	7/9/2019	17:								
6	7/9/2019	17:								
7	7/9/2019	17:								
8	7/9/2019	17:								
9	7/9/2019	17:								
10	7/9/2019	17								
11	7/9/2019	17:								
12	7/9/2019	17:								
13	7/9/2019	17:								
14	7/9/2019	17:								
15	7/9/2019	17:								
16	7/9/2019	17:								
17	7/9/2019	17:								
18	7/9/2019	17:54:24	Unit1	31	30.88	-0.12	0	33.57	13.16%	Heat

Average: 21.56222059 Count: 2790 Sum: 60093.90877 100%

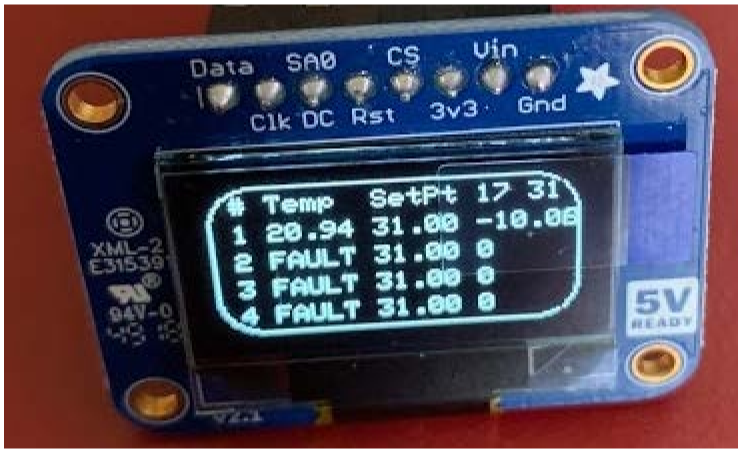
Alternatively, if you want to analyze all 4 of the Units you can copy the Temp column for each of the Units, parallel to the column with Unit1 temperatures.

FAULT Error Codes

The FAULT Error Codes are displayed in the window with the OLED (128x64).

Threshold Faults:		
HIGHTHRESH	RTD High Threshold	128
LOWTHRESH	RTD Low Threshold	64
Hardware Failures:		
REFINLOW	REFIN- > 0.85 x Bias	32
REFINHIGH	REFIN- < 0.85 x Bias - FORCE- open	16
RTDINLOW	RTDIN- < 0.85 x Bias - FORCE- open	8
OVUV	Under/Over voltage	4
Sensor Communication Failure:		
NOSENSOR	Unable to connect to sensor	0

Sample error display with only one sensor connected and FAULT Error Code is 0.



```
# Temp  SetPt 17 31
1 20.94 31.00 -10.08
2 FAULT 31.00 0
3 FAULT 31.00 0
4 FAULT 31.00 0
```

Updating the Arduino INO file

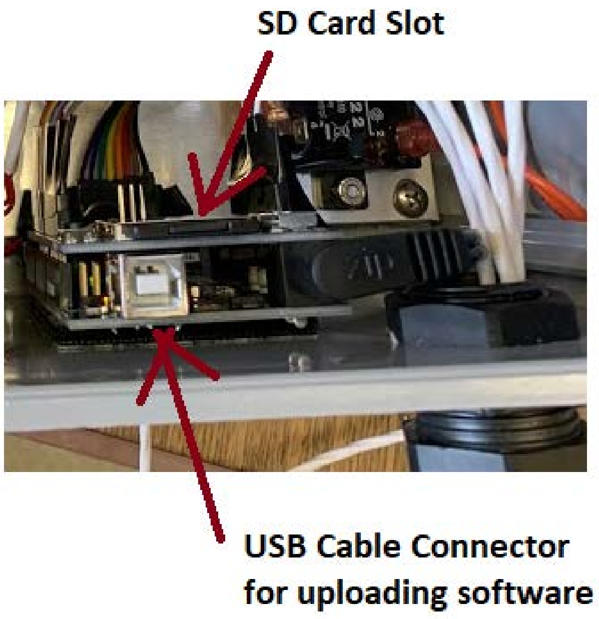
The software for the Controller can be updated by means of a USB cable between a laptop computer and the Arduino Controller, and the Arduino IDE (Integrated Development Environment).

The Arduino IDE can be downloaded here for Windows / Mac:
 https://www.arduino.cc/en/main/software

After installing the Arduino IDE the following libraries must be installed:

Tools -> Manage Libraries -> Install

`PID`	`PID by Brett Beauregard Version 1.2.0`
`Adafruit_MAX31865`	`Adafruit MAX31865 library by Adafruit Version 1.0.1`
`SPI`	`Built-In`
`Wire`	`Built-In`
`Adafruit_GFX_Library`	`Adafruit GFX Library by Adafruit Version 1.4.2`
`Adafruit_SSD1306`	`Adafruit SSD1306 by Adafruit Version 1.2.9`
`RTClib`	`Built-In`
`SD`	`Built-In`



The process for updating the Arduino Controller is as follows:

- Copy the 4_Unit_PID_Controller_SDCard.ino file into the folder:
  Documents\Arduino\4_Unit_PID_Controller_SDCard

- Sketch -> Verify/Compile
  To confirm that the new INO file will compile correctly. Note that the first time this is done on a newly installed Arduino IDE, you will likely need to add some missing libraries (MAX31865, etc) see above listing of required Libraries.

- Connect the USB cable from the laptop to the Arduino

- Select the port used by the USB connection
  Tools -> Port -> xxx   (e.g. COM5)

- Sketch -> Upload
  The INO file will be compiled, uploaded to the Arduino, and execution will begin

- Disconnect the USB cable

- Alternatively for additional Debug Output the Serial Monitor can be opened to display debug output

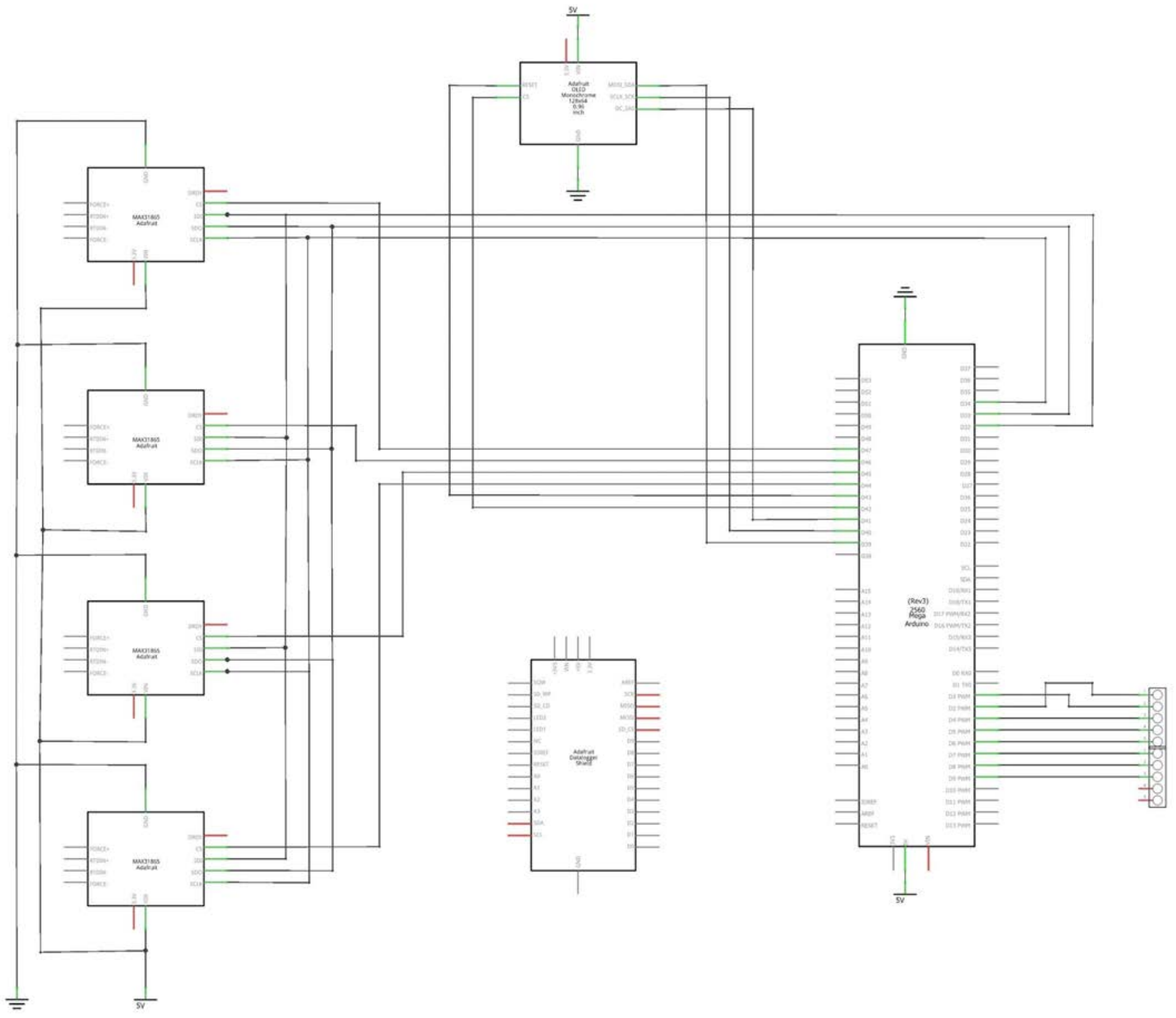
Formatting SD Card under Windows/Mac

Chances a new SD Card is already pre-formatted with a FAT filesystem. However you may have problems with how the factory formats the card, or if it's an old card it needs to be reformatted. The Arduino SD library used supports both FAT16 and FAT32 file systems. Note that formatting will erase the card so save anything you want first.

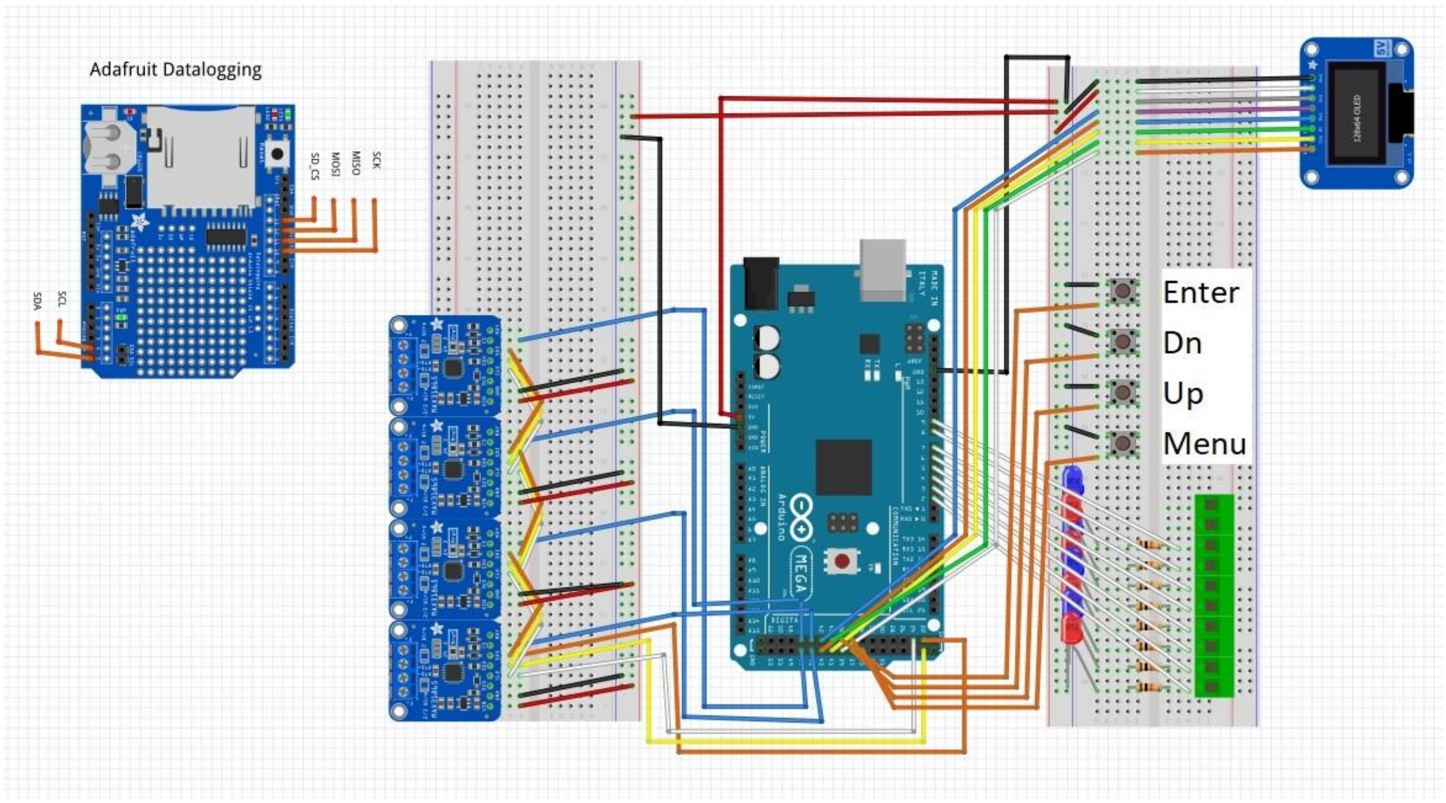
The official SD formatter is available from https://www.sdcard.org/downloads/formatter_4/

Download it and run it on your computer, there's also a manual linked from that page for use.

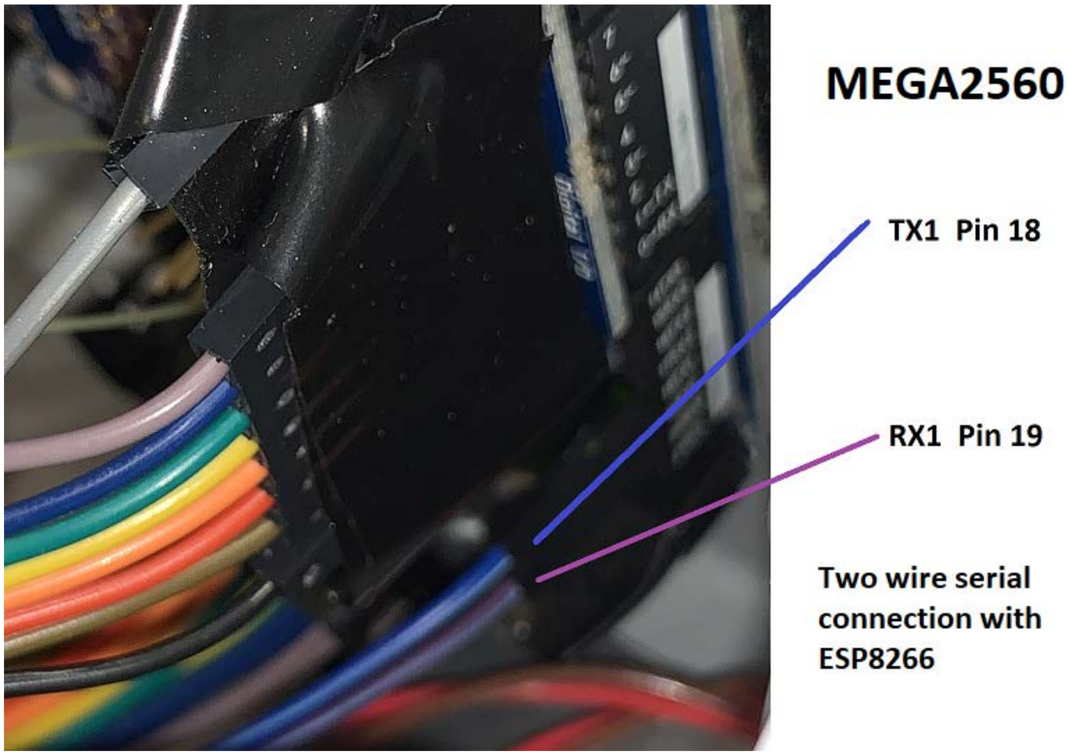
4 Port PID Controller Schematic

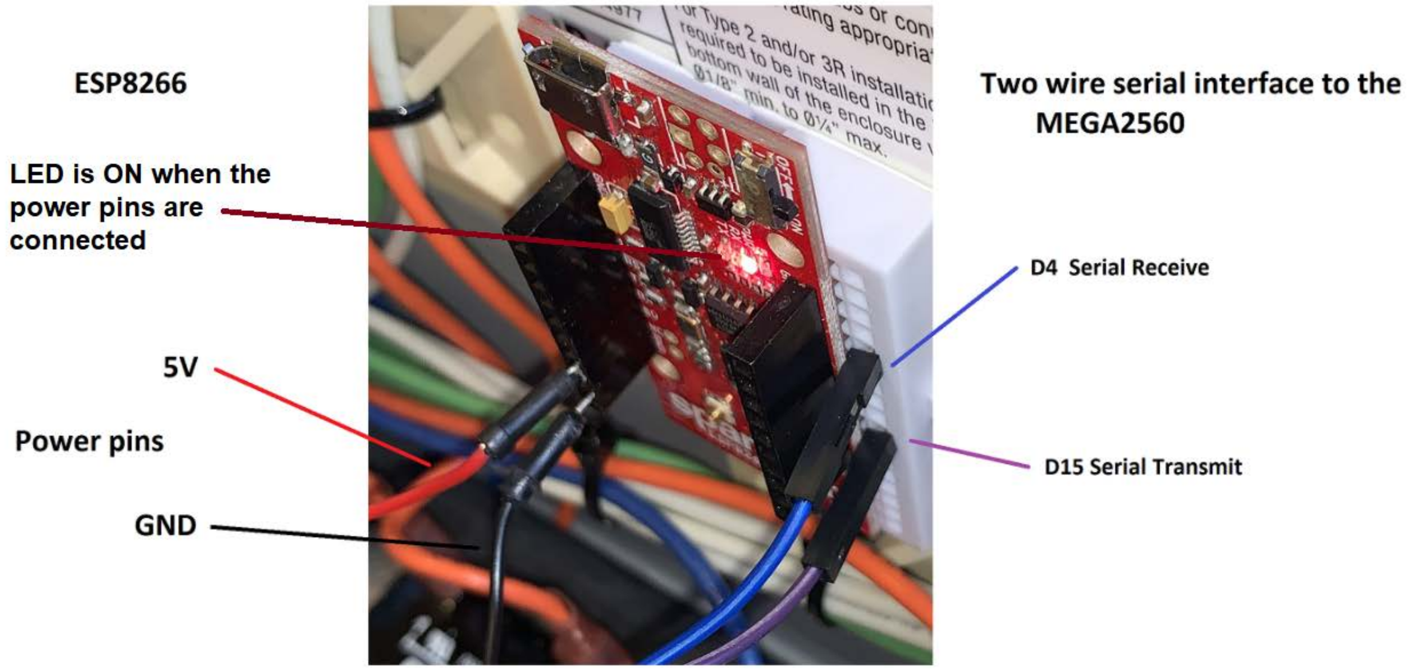


4 Port PID Controller Breadboard



ESP8266 WiFi Access Point Connections




ESP8266
LED is ON when the power pins are connected
5V
Power pins
GND
Two wire serial interface to the MEGA2560
D4 Serial Receive
D15 Serial Transmit

Controller INO Source Code

```
/***********************************************************************
  4 Unit PID Controller

  Original Code:  2019-07-08

  Tom Rolander, MSEE
  Mentor, Circuit Design & Software
  Miller Library, Fabrication Lab
  Hopkins Marine Station, Stanford University,
  120 Ocean View Blvd, Pacific Grove, CA 93950
  +1 831.915.9526 | rolander@stanford.edu

 ***********************************************************************/

#define VERSION "Ver 0.9 2021-10-11"

#define DEBUGGING 1

int maxRTD=4;

#define DELAY_DIVISOR 16    // compensate for the change of frequency for Timer 0

#define DELAY_BETWEEN_UPDATES 10000
#define DELAY_BETWEEN_LOGGING 60000

#define MINIMUM_COOL  60000

#define MINIMUM_VALID_TEMP  10.0
#define MAXIMUM_VALID_TEMP  50.0

int iCoolUpdates[4] = {0,0,0,0};

unsigned long timeLastPID = 0;
unsigned long timeLastLog = 0;

//#include <MemoryFree.h>

#include <PID_v1.h>

//Define Variables we'll be connecting to with the PID library
double Setpoint[4] = {31.0, 31.0, 31.0, 31.0};
double Input[4] = {0.0, 0.0, 0.0, 0.0};
double Output[4] = {0.0, 0.0, 0.0, 0.0};

double SetpointNew;

double prevTemp[4] = {0.0, 0.0, 0.0, 0.0};
double prevSetpoint[4] = {31.0, 31.0, 31.0, 31.0};

//double offsetTemp[4] = {-0.16, 0.06, -0.01, 0.12};
double offsetTemp[4] = {0.0, 0.0, 0.0, 0.0};

double Kp = 2;
double Ki = 5;
double Kd = 1;

// From: https://www.diva-portal.org/smash/get/diva2:678519/FULLTEXT01.pdf
#if 0
double Kp = 14.4;
double Ki = 6;
double Kd = 1.5;
#endif

int POn = P_ON_E;
int Direction[4] = {DIRECT, DIRECT, DIRECT, DIRECT};
double MinPctDutyCycle[4] = {10.0,10.0,10.0,10.0};
double MaxPctDutyCycle[4] = {50.0,50.0,50.0,50.0};

//Specify the links and initial tuning parameters
#if 0
PID myPID(&Input, &Output, &Setpoint, Kp, Ki, Kd, P_ON_E, DIRECT); //P_ON_M specifies that
Proportional on Measurement be used
                                                                    //P_ON_E (Proportional on Error)
is the default behavior
#endif

PID myPID[4] = {
  PID(&Input[0], &Output[0], &Setpoint[0], Kp, Ki, Kd, POn, Direction[0]),
  PID(&Input[1], &Output[1], &Setpoint[1], Kp, Ki, Kd, POn, Direction[1]),
  PID(&Input[2], &Output[2], &Setpoint[2], Kp, Ki, Kd, POn, Direction[2]),
  PID(&Input[3], &Output[3], &Setpoint[3], Kp, Ki, Kd, POn, Direction[3])
};

unsigned int Setpoints_Thousandths[4][240];

double DeltaIncrement = 0.5;

#include <Adafruit_MAX31865.h>

// Use software SPI: CS, DI, DO, CLK
Adafruit_MAX31865 max[4] = {
  Adafruit_MAX31865(44, 32, 33, 34),
  Adafruit_MAX31865(45, 32, 33, 34),
  Adafruit_MAX31865(46, 32, 33, 34),
  Adafruit_MAX31865(47, 32, 33, 34)
};

// The value of the Rref resistor. Use 430.0 for PT100 and 4300.0 for PT1000
#define RREF      430.0
// The 'nominal' 0-degrees-C resistance of the sensor
// 100.0 for PT100, 1000.0 for PT1000
#define RNOMINAL  100.0

#include <SPI.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128 // OLED display width, in pixels
```

```
#define SCREEN_HEIGHT 64 // OLED display height, in pixels

// Declaration for SSD1306 display connected using software SPI (default case):
#define OLED_MOSI  39
#define OLED_CLK   40
#define OLED_DC    41
#define OLED_CS    42
#define OLED_RESET 43
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT,
  OLED_MOSI, OLED_CLK, OLED_DC, OLED_RESET, OLED_CS);

/* Comment out above, uncomment this block to use hardware SPI
#define OLED_DC     6
#define OLED_CS     7
#define OLED_RESET  8
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT,
  &SPI, OLED_DC, OLED_RESET, OLED_CS);
*/

int lineSpacing=12;
int xOffset = 8;
int yOffset = 4;

// Date and time functions using a DS1307 RTC connected via I2C and Wire lib
#include "RTClib.h"

RTC_PCF8523 rtc;
DateTime now;

int counter = 0;

int prevHour = 0;
int prevMin = 0;


// SD Card used for data logging
#include <SD.h>

File fileSDCard;

// SD Shield
//
// change this to match your SD shield or module;
// Arduino Ethernet shield: pin 4
// Adafruit SD shields and modules: pin 10
// Sparkfun SD shield: pin 8
// MKRZero SD: SDCARD_SS_PIN
#define chipSelectSDCard 10

#if 0
char cEncodedBuffer[1024 + 64];
char *cDecodedBuffer = cEncodedBuffer;
//char cDecodedBuffer[1024 + 64];
char *pLogging = &cEncodedBuffer[0];
#endif
```

```
static int bFileUploading = false;
//static char sFilename[16] = "";
//static char sFilenameBak[16] = "";

bool bSDLogFail = false;
int  iToggle = 0;

#define HeaterUnit1 2
#define CoolerUnit1 3
#define HeaterUnit2 4
#define CoolerUnit2 5
#define HeaterUnit3 6
#define CoolerUnit3 7
#define HeaterUnit4 8
#define CoolerUnit4 9

#define menuPin   35
#define upPin     36
#define dnPin     37
#define enterPin  38

#define STATE_RUN                 0
#define STATE_SP_WAIT             1
#define STATE_SP                  2
#define STATE_SP_ENTER_WAIT       3
#define STATE_SP_ENTER            4
#define STATE_SP_ENTER_UP_WAIT    5
#define STATE_SP_ENTER_DN_WAIT    6
#define STATE_SP_UPDATE_WAIT      7
#define STATE_CFG_WAIT            8
#define STATE_CFG                 9
#define STATE_CFG_ENTER_WAIT      10
#define STATE_CFG_ENTER           11
#define STATE_CFG_ENTER_UP_WAIT   12
#define STATE_CFG_ENTER_DN_WAIT   13
#define STATE_CFG_UPDATE_WAIT     14
#define STATE_RUN_WAIT            15
#define STATE_HI_PEAK_WAIT        16
#define STATE_HI_PEAK             17
#define STATE_HI_PEAK_FINISH      18
#define STATE_LO_VALY_WAIT        19
#define STATE_LO_VALY             20
#define STATE_LO_VALY_FINISH      21
#define STATE_STBY                22

int HeaterUnits[4] = {HeaterUnit1, HeaterUnit2, HeaterUnit3, HeaterUnit4};
int CoolerUnits[4] = {CoolerUnit1, CoolerUnit2, CoolerUnit3, CoolerUnit4};

int currentState = STATE_RUN;
int currentSetpoint = 0;
int lastState = 0;
int savedState;

int updatingSetpoint = false;

int previousEnterButton = false;
int menuButton;
int upButton;
int dnButton;
int enterButton;

int menuButtonLast = HIGH;
int upButtonLast = HIGH;
int dnButtonLast = HIGH;
int enterButtonLast = HIGH;

#define BUTTON_HOLD_TIME 3000
#define BUTTON_HOLD_DELTA 200

int buttonDownStartTime = 0;
int buttonDownCurrentTime = 0;
int buttonDownIncrementTime = 0;

void(* resetFunc) (void) = 0;

void setup()
{
  Wire.begin();

  Serial.begin(9600);
  Serial.println(F(""));
  Serial.println(VERSION);
  Serial.println(F("Serial Initialized..."));

  Serial1.begin(19200);
//  Serial1.setTimeout(10000);

#if 0
  for (int i=0; i< 4; i++)
  {
    pinMode(HeaterUnits[i], OUTPUT);
    //digitalWrite(HeaterUnits[i], HIGH);
    analogWrite(HeaterUnits[i], 255.0);
    pinMode(CoolerUnits[i], OUTPUT);
  }
  Serial.println(F("SYSTEM HALTED!"));
  while (1);
#endif

  for (int i=0; i< 4; i++)
  {
    for (int j=0; j<240; j++)
      Setpoints_Thousandths[i][j] = 31000;
  }

//For Arduino Mega1280, Mega2560, MegaADK, Spider or any other board using ATmega1280 or ATmega2560

//---------------------------------------------- Set PWM frequency for D4 & D13
------------------------------
```

```
//TCCR0B = TCCR0B & B11111000 | B00000001;    // set timer 0 divisor to     1 for PWM frequency of
62500.00 Hz
//TCCR0B = TCCR0B & B11111000 | B00000010;    // set timer 0 divisor to     8 for PWM frequency of
7812.50 Hz
//TCCR0B = TCCR0B & B11111000 | B00000011;    // set timer 0 divisor to    64 for PWM frequency of
976.56 Hz
//TCCR0B = TCCR0B & B11111000 | B00000100;    // set timer 0 divisor to   256 for PWM frequency of
244.14 Hz
TCCR0B = TCCR0B & B11111000 | B00000101;    // set timer 0 divisor to  1024 for PWM frequency of
61.04 Hz


//---------------------------------------------- Set PWM frequency for D11 & D12
-----------------------------

//TCCR1B = TCCR1B & B11111000 | B00000001;    // set timer 1 divisor to     1 for PWM frequency of
31372.55 Hz
//TCCR1B = TCCR1B & B11111000 | B00000010;    // set timer 1 divisor to     8 for PWM frequency of
3921.16 Hz
//TCCR1B = TCCR1B & B11111000 | B00000011;    // set timer 1 divisor to    64 for PWM frequency of
490.20 Hz
//TCCR1B = TCCR1B & B11111000 | B00000100;    // set timer 1 divisor to   256 for PWM frequency of
122.55 Hz
//TCCR1B = TCCR1B & B11111000 | B00000101;    // set timer 1 divisor to  1024 for PWM frequency of
30.64 Hz

//---------------------------------------------- Set PWM frequency for D9 & D10
------------------------------

//TCCR2B = TCCR2B & B11111000 | B00000001;    // set timer 2 divisor to     1 for PWM frequency of
31372.55 Hz
//TCCR2B = TCCR2B & B11111000 | B00000010;    // set timer 2 divisor to     8 for PWM frequency of
3921.16 Hz
//TCCR2B = TCCR2B & B11111000 | B00000011;    // set timer 2 divisor to    32 for PWM frequency of
980.39 Hz
//TCCR2B = TCCR2B & B11111000 | B00000100;    // set timer 2 divisor to    64 for PWM frequency of
490.20 Hz
//TCCR2B = TCCR2B & B11111000 | B00000101;    // set timer 2 divisor to   128 for PWM frequency of
245.10 Hz
//TCCR2B = TCCR2B & B11111000 | B00000110;    // set timer 2 divisor to   256 for PWM frequency of
122.55 Hz
  TCCR2B = TCCR2B & B11111000 | B00000111;    // set timer 2 divisor to  1024 for PWM frequency of
30.64 Hz


//---------------------------------------------- Set PWM frequency for D2, D3 & D5
---------------------------

//TCCR3B = TCCR3B & B11111000 | B00000001;    // set timer 3 divisor to     1 for PWM frequency of
31372.55 Hz
//TCCR3B = TCCR3B & B11111000 | B00000010;    // set timer 3 divisor to     8 for PWM frequency of
3921.16 Hz
//TCCR3B = TCCR3B & B11111000 | B00000011;    // set timer 3 divisor to    64 for PWM frequency of
490.20 Hz
//TCCR3B = TCCR3B & B11111000 | B00000100;    // set timer 3 divisor to   256 for PWM frequency of
122.55 Hz
```

```
  TCCR3B = TCCR3B & B11111000 | B00000101;    // set timer 3 divisor to  1024 for PWM frequency of
30.64 Hz


//---------------------------------------------- Set PWM frequency for D6, D7 & D8
---------------------------

//TCCR4B = TCCR4B & B11111000 | B00000001;    // set timer 4 divisor to     1 for PWM frequency of
31372.55 Hz
//TCCR4B = TCCR4B & B11111000 | B00000010;    // set timer 4 divisor to     8 for PWM frequency of
3921.16 Hz
//TCCR4B = TCCR4B & B11111000 | B00000011;    // set timer 4 divisor to    64 for PWM frequency of
490.20 Hz
//TCCR4B = TCCR4B & B11111000 | B00000100;    // set timer 4 divisor to   256 for PWM frequency of
122.55 Hz
  TCCR4B = TCCR4B & B11111000 | B00000101;    // set timer 4 divisor to  1024 for PWM frequency of
30.64 Hz


//---------------------------------------------- Set PWM frequency for D44, D45 & D46
------------------------

//TCCR5B = TCCR5B & B11111000 | B00000001;    // set timer 5 divisor to     1 for PWM frequency of
31372.55 Hz
//TCCR5B = TCCR5B & B11111000 | B00000010;    // set timer 5 divisor to     8 for PWM frequency of
3921.16 Hz
//TCCR5B = TCCR5B & B11111000 | B00000011;    // set timer 5 divisor to    64 for PWM frequency of
490.20 Hz
//TCCR5B = TCCR5B & B11111000 | B00000100;    // set timer 5 divisor to   256 for PWM frequency of
122.55 Hz
//TCCR5B = TCCR5B & B11111000 | B00000101;    // set timer 5 divisor to  1024 for PWM frequency of
30.64 Hz


  for (int i=0; i<maxRTD; i++) {
    max[i].begin(MAX31865_3WIRE);  // set to 2WIRE or 4WIRE as necessary
  }

  pinMode(menuPin, INPUT_PULLUP);
  pinMode(upPin, INPUT_PULLUP);
  pinMode(dnPin, INPUT_PULLUP);
  pinMode(enterPin, INPUT_PULLUP);

  // SSD1306_SWITCHCAPVCC = generate display voltage from 3.3V internally
  if(!display.begin(SSD1306_SWITCHCAPVCC)) {
    Serial.println(F("SSD1306 allocation failed"));
    for(;;); // Don't proceed, loop forever
  }

  displayFrame();
  display.setTextSize(1);      // Normal 1:1 pixel scale
  display.setTextColor(WHITE); // Draw white text
  display.cp437(true);         // Use full 256 char 'Code Page 437' font

  display.setCursor(xOffset, yOffset+0);
  display.print(F("4 Unit PID"));
  display.setCursor(xOffset, yOffset+lineSpacing);
  display.print(F("Hopkins 4xTomPort"));
  display.setCursor(xOffset, yOffset+(2*lineSpacing));
  display.print(VERSION);
  display.display();

  delay(5000/DELAY_DIVISOR);

  SetupSDCardOperations();

// Initialize the Real Time Clock
  if (! rtc.begin())
  {
    displayFrame();
    display.setCursor(xOffset, yOffset+(1*lineSpacing));
    display.print(F("*** ERROR ***   "));
    display.setCursor(xOffset, yOffset+(2*lineSpacing));
    display.print(F("Couldnt find RTC"));
    while (1);
  }
  if (! rtc.initialized())
  {
    displayFrame();
    display.setCursor(xOffset, yOffset+(1*lineSpacing));
    display.print(F("*** WARN ***    "));
    display.setCursor(xOffset, yOffset+(2*lineSpacing));
    display.print(F("RTC isnt running"));

    // following line sets the RTC to the date & time this sketch was compiled
    rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
    // This line sets the RTC with an explicit date & time, for example to set
    // January 21, 2014 at 3am you would call:
    // rtc.adjust(DateTime(2014, 1, 21, 3, 0, 0));
  }
  now = rtc.now();

  Serial.println(F("RTC Time:"));
  Serial.print(now.year(), DEC);
  Serial.print(F("/"));
  Serial.print(now.month(), DEC);
  Serial.print(F("/"));
  Serial.print(now.day(), DEC);
  Serial.print(F(" "));
  if (now.hour() < 10) Serial.print(F("0"));
  Serial.print(now.hour(), DEC);
  Serial.print(F(":"));
  if (now.minute() < 10) Serial.print(F("0"));
  Serial.print(now.minute(), DEC);
  Serial.print(F(":"));
  if (now.second() < 10) Serial.print(F("0"));
  Serial.print(now.second(), DEC);
  Serial.println(F(" "));


  displayFrame();
  display.setCursor(xOffset, yOffset+(1*lineSpacing));
  display.print(F("*** DATE ***    "));
  display.setCursor(xOffset, yOffset+(2*lineSpacing));
  display.print(now.year(), DEC);
  display.print(F("/"));
  OledDisplayPrintTwoDigits(now.month());
  display.print(F("/"));
  OledDisplayPrintTwoDigits(now.day());
  display.print(F(" "));
  OledDisplayPrintTwoDigits(now.hour());
  display.print(F(":"));
  OledDisplayPrintTwoDigits(now.minute());
  display.display();
  delay(2000/DELAY_DIVISOR);

  Serial.println(F("Tuning Parameters"));
  Serial.print(F(" Kp = "));
  Serial.println(Kp);
  Serial.print(F(" Ki = "));
  Serial.println(Ki);
  Serial.print(F(" Kd = "));
  Serial.println(Kd);
  Serial.print(F(" Prop on "));
  if (POn == P_ON_E)
    Serial.println(F("Error"));
  else
    Serial.println(F("Measure"));

  displayFrame();
  display.setCursor(xOffset, yOffset+(0*lineSpacing));
  display.print(F("Tuning Parameters"));
  display.setCursor(xOffset, yOffset+(1*lineSpacing));
  display.print(F(" Kp = "));
  display.print(Kp);
  display.setCursor(xOffset, yOffset+(2*lineSpacing));
  display.print(F(" Ki = "));
  display.print(Ki);
  display.setCursor(xOffset, yOffset+(3*lineSpacing));
  display.print(F(" Kd = "));
  display.print(Kd);
  display.setCursor(xOffset, yOffset+(4*lineSpacing));
  display.print(F(" Prop on "));
  if (POn == P_ON_E)
    display.print(F("Error"));
  else
    display.print(F("Measure"));
  display.display();
  delay(2000/DELAY_DIVISOR);

  Serial.println(F("Offsets"));
  for (int i=0; i<4; i++)
  {
    Serial.print(F(" Offset"));
    Serial.print(i+1);
    Serial.print(F(" = "));
    Serial.println(offsetTemp[i]);
  }
```

```
displayFrame();
display.setCursor(xOffset, yOffset+(0*lineSpacing));
display.print(F("Offsets"));
for (int i=0; i<4; i++)
{
  display.setCursor(xOffset, yOffset+((i+1)*lineSpacing));
  display.print(F(" Offset"));
  display.print(i+1);
  display.print(F(" = "));
  display.print(offsetTemp[i]);
}
display.display();
delay(10000/DELAY_DIVISOR);

Serial.println(F("MinPcts"));
for (int i=0; i<4; i++)
{
  Serial.print(F(" MinPcts"));
  Serial.print(i+1);
  Serial.print(F(" = "));
  Serial.println((MinPctDutyCycle[i]*100)/255);
}

displayFrame();
display.setCursor(xOffset, yOffset+(0*lineSpacing));
display.print(F("MinPcts"));
for (int i=0; i<4; i++)
{
  display.setCursor(xOffset, yOffset+((i+1)*lineSpacing));
  display.print(F(" MinPcts"));
  display.print(i+1);
  display.print(F(" = "));
  display.print((MinPctDutyCycle[i]*100)/255);
}
display.display();
delay(2000/DELAY_DIVISOR);

Serial.println(F("MaxPcts"));
for (int i=0; i<4; i++)
{
  Serial.print(F(" MaxPcts"));
  Serial.print(i+1);
  Serial.print(F(" = "));
  Serial.println((MaxPctDutyCycle[i]*100)/255);
}

displayFrame();
display.setCursor(xOffset, yOffset+(0*lineSpacing));
display.print(F("MaxPcts"));
for (int i=0; i<4; i++)
{
  display.setCursor(xOffset, yOffset+((i+1)*lineSpacing));
  display.print(F(" MaxPcts"));
  display.print(i+1);
  display.print(F(" = "));
```

```
    display.print((MaxPctDutyCycle[i]*100)/255);
  }
  display.display();
  delay(2000/DELAY_DIVISOR);

  displayRun();

  Serial.println(F("Min and Max Limits"));

  //turn the PID on
  for (int i=0; i<4; i++)
  {
    //initialize the variables we're linked to

    Serial.print(i+1);
    Serial.print(F(" "));
    Serial.print(F(" MinLimit"));
    Serial.print(F(" = "));
    Serial.print(MinPctDutyCycle[i]);
    Serial.print(F(" "));
    Serial.print(F(" MaxLimit"));
    Serial.print(F(" = "));
    Serial.println(MaxPctDutyCycle[i]);

    Input[i] = 31.0;
    Setpoint[i] = 31.0;
    myPID[i].SetMode(AUTOMATIC);
    myPID[i].SetOutputLimits(MinPctDutyCycle[i],MaxPctDutyCycle[i]);
    Direction[i] = DIRECT;
    myPID[i].SetControllerDirection(Direction[i]);
  }
}

void displayRun()
{
  displayFrame();
  for(int i=0; i<1; i++) {
  display.drawRoundRect(i, i, display.width()-2*i, display.height()-2*i,
      display.height()/4, WHITE);
  }

  display.setCursor(xOffset, yOffset+0);
  display.print(F("# Temp  SetPt"));

  for (int i=1; i<5; i++)
  {
    display.setCursor(xOffset, yOffset+(i*lineSpacing));
    display.print(i);
    display.print(F("       "));
    display.print(Setpoint[i-1]);
  }
  display.display();
}

void loop()
{
  FileTransfer();
  if (bFileUploading)
    return;

  now = rtc.now();
  int currentHour = now.hour();
  int currentMin  = now.minute();
  int currentSec  = now.second();

  unsigned long timeCurrent = millis() * DELAY_DIVISOR;

  double temp[4] = {0.0, 0.0, 0.0, 0.0};
  double delta[4] = {0.0, 0.0, 0.0, 0.0};
  uint8_t fault[4] = {false, false, false, false};

  menuButton = digitalRead(menuPin);
  upButton = digitalRead(upPin);
  dnButton = digitalRead(dnPin);
  enterButton = digitalRead(enterPin);

  switch (currentState)
  {
    case STATE_STBY:
      break;

    case STATE_RUN:
      if (menuButton == LOW)
      {
        previousEnterButton = false;
        currentState = STATE_SP_WAIT;
        currentSetpoint = 0;
        break;
      }

//Serial.print(timeCurrent); Serial.print(F(" ")); Serial.print(timeLastPID); Serial.print(F(" "));
Serial.print(timeCurrent-timeLastPID); Serial.print(F(" ")); Serial.println(DELAY_BETWEEN_UPDATES);
      if ((timeCurrent - timeLastPID) < DELAY_BETWEEN_UPDATES)
      {
        char separator;
        counter++;
        if ((counter & 1) == 1)
          separator = ':';
        else
          separator = ' ';

        if (prevHour != currentHour || prevMin != currentMin)
        {
          display.fillRect(xOffset+(14*(5+1)), yOffset+0,5*(5+1),8,BLACK);
          display.setCursor(xOffset+(14*(5+1)), yOffset+0);
          OledDisplayPrintTwoDigits(now.hour());
          display.print(separator);
          OledDisplayPrintTwoDigits(now.minute());
          prevHour = currentHour;
          prevMin = currentMin;
        }
        else
        {
          display.fillRect(xOffset+(16*(5+1)), yOffset+0,1*(5+1),8,BLACK);
          display.setCursor(xOffset+(16*(5+1)), yOffset+0);
          display.print(separator);
        }
        display.display();

        delay(1000/DELAY_DIVISOR);
        return;
      }

      timeLastPID = timeCurrent;
      int bSetTimeCurrent = false;

      for (int i=0; i<maxRTD; i++)
      {
        char szUnit[] = "Unit?";
        szUnit[4] = char ('1'+i);

        // Update the Setpoint from the table
        int index = (currentHour*10) + ((currentMin*10)/60);
        Setpoint[i] = double(Setpoints_Thousandths[i][index]) / 1000.0;
        //Serial.print(F("Setpoint: "));
        //Serial.println(Setpoint[i],3);

        if (Setpoint[i] != prevSetpoint[i])
        {
          display.fillRect(xOffset+(8*(5+1)), yOffset+((i+1)*lineSpacing),5*(5+1),8,BLACK);
          display.setCursor(xOffset+(8*(5+1)), yOffset+((i+1)*lineSpacing));
          display.print(Setpoint[i]);
          prevTemp[i] = 0.0;  // force temp update
          prevSetpoint[i] = Setpoint[i];
        }

        uint16_t rtd = max[i].readRTD();

      //  Serial.print(F("RTD value: ")); Serial.println(rtd);
        float ratio = rtd;
        ratio /= 32768;
      //  Serial.print(F("Ratio = ")); Serial.println(ratio,8);
      //  Serial.print(F("Resistance = ")); Serial.println(RREF*ratio,8);

        Serial.print(now.year(), DEC);
        Serial.print(F("/"));
        Serial.print(now.month(), DEC);
        Serial.print(F("/"));
        Serial.print(now.day(), DEC);
        Serial.print(F(" "));
        if (now.hour() < 10) Serial.print(F("0"));
        Serial.print(now.hour(), DEC);
        Serial.print(F(":"));
        if (now.minute() < 10) Serial.print(F("0"));
        Serial.print(now.minute(), DEC);
        Serial.print(F(":"));
        if (now.second() < 10) Serial.print(F("0"));
        Serial.print(now.second(), DEC);
        Serial.print(F(" "));

        temp[i] = max[i].temperature(RNOMINAL, RREF);

        temp[i] += offsetTemp[i];

        // Check and print any faults
        fault[i] = max[i].readFault();
        if (fault[i])
        {
          Serial.print(i+1); Serial.print(F(" Fault 0x")); Serial.print(fault[i], HEX);
          if (fault[i] & MAX31865_FAULT_HIGHTHRESH) {
            Serial.println(F(" RTD High Threshold"));
          }
          if (fault[i] & MAX31865_FAULT_LOWTHRESH) {
            Serial.println(F(" RTD Low Threshold"));
          }
          if (fault[i] & MAX31865_FAULT_REFINLOW) {
            Serial.println(F(" REFIN- > 0.85 x Bias"));
          }
          if (fault[i] & MAX31865_FAULT_REFINHIGH) {
            Serial.println(F(" REFIN- < 0.85 x Bias - FORCE- open"));
          }
          if (fault[i] & MAX31865_FAULT_RTDINLOW) {
            Serial.println(F(" RTDIN- < 0.85 x Bias - FORCE- open"));
          }
          if (fault[i] & MAX31865_FAULT_OVUV) {
            Serial.println(F(" Under/Over voltage"));
          }
          max[i].clearFault();
        }
        else
        if ((temp[i] < MINIMUM_VALID_TEMP) || (temp[i] > MAXIMUM_VALID_TEMP))
        {
          Serial.print(i+1); Serial.print(F(" Temp = ")); Serial.print(temp[i]);
          Serial.println(F(" INVALID"));
        }
        else
        {
          Serial.print(i+1); Serial.print(F(" Temp = ")); Serial.print(temp[i]);
          Serial.print(F(", Delta = ")); Serial.print(temp[i] - Setpoint[i]);
          Serial.print(F(", Offset = ")); Serial.print(offsetTemp[i]);
        }

        if (fault[i]  || (temp[i] < MINIMUM_VALID_TEMP) || (temp[i] > MAXIMUM_VALID_TEMP))
        {
          if ((timeCurrent - timeLastLog) >= DELAY_BETWEEN_LOGGING)
          {
            bSetTimeCurrent = true;
            SDLogging(szUnit, Setpoint[i], 0, fault[i], 0, 0, "FAULT");

            display.fillRect(xOffset+(2*(5+1)), yOffset+((i+1)*lineSpacing),5*(5+1),8,BLACK);
            display.setCursor(xOffset+(2*(5+1)), yOffset+((i+1)*lineSpacing));
            display.print(F("FAULT"));

            display.fillRect(xOffset+(14*(5+1)), yOffset+((i+1)*lineSpacing),5*(5+1),8,BLACK);
```

```
        display.setCursor(xOffset+(14*(5+1)), yOffset+((i+1)*lineSpacing));

        if (fault[i])
        {
          display.print(fault[i]);
          SDLogging(szUnit, Setpoint[i], -1, fault[i], 0, 0, "FAULT");
        }
        else
        {
          display.print(0);
          SDLogging(szUnit, Setpoint[i], 0, -1, 0, 0, "FAULT");
        }
      }
    }
    else
    {
      Input[i] = temp[i];

      char *strHeatingOrCooling;

      if (iCoolUpdates[i] > 0)
      {
        iCoolUpdates[i]++;
        Output[i] = 255.0;
        if (iCoolUpdates[i] >= (MINIMUM_COOL/DELAY_BETWEEN_UPDATES))
        {
          iCoolUpdates[i] = 0;
          strHeatingOrCooling = "COOL (completed)";
        }
        else
        {
          strHeatingOrCooling = "COOL (waiting)";
        }
      }
      else
      {
        if (Input[i] >= Setpoint[i])
        {
          // Cooling
          Direction[i] = REVERSE;
          myPID[i].SetControllerDirection(Direction[i]);
          myPID[i].Compute();
          if (Input[i] > Setpoint[i] + 0.1)
          {
            // Turn on COOLER
            iCoolUpdates[i] = 1;
            strHeatingOrCooling = "Cool";
            Output[i] = 255.0;  // Force 100% Duty Cycle for cooling
            analogWrite(HeaterUnits[i],0.0);
            analogWrite(CoolerUnits[i],Output[i]);
          }
          else
          {
            // Turn off COOLER and HEATER
            strHeatingOrCooling = "OFF";
            Output[i] = 0.0;
```

```
              analogWrite(HeaterUnits[i],Output[i]);
              analogWrite(CoolerUnits[i],Output[i]);
            }
          }
          else
          {
            // Heating
            Direction[i] = DIRECT;
            myPID[i].SetControllerDirection(Direction[i]);
            myPID[i].Compute();
            strHeatingOrCooling = "Heat";
            analogWrite(HeaterUnits[i],Output[i]);
            analogWrite(CoolerUnits[i],0.0);
            iCoolUpdates[i] = 0;
          }

        }

        Serial.print(F(", Setpoint = ")); Serial.print(Setpoint[i]);
        double DutyCycle = (Output[i]/255.0)*100;
        Serial.print(F(", DutyCycle = ")); Serial.print(DutyCycle); Serial.print(F("%"));
        Serial.print(F(", "));
        Serial.println(strHeatingOrCooling);

        if ((timeCurrent - timeLastLog) >= DELAY_BETWEEN_LOGGING)
        {
          if (i+1 >= maxRTD)
            timeLastLog = timeCurrent;
          SDLogging(szUnit, Setpoint[i], temp[i], (temp[i] - Setpoint[i]), offsetTemp[i],
Output[i], strHeatingOrCooling);
        }

        if (temp[i] != prevTemp[i])
        {
          prevTemp[i] = temp[i];
          display.fillRect(xOffset+(2*(5+1)), yOffset+((i+1)*lineSpacing),5*(5+1),8,BLACK);
          display.setCursor(xOffset+(2*(5+1)), yOffset+((i+1)*lineSpacing));
          display.print(temp[i]);

          delta[i] = temp[i] - Setpoint[i];
          display.fillRect(xOffset+(14*(5+1)), yOffset+((i+1)*lineSpacing),5*(5+1),8,BLACK);
          display.setCursor(xOffset+(14*(5+1)), yOffset+((i+1)*lineSpacing));
          if (delta[i] >= 0.0)
            display.print('+');
          display.print(delta[i]);
        }

     }
    }
    if (bSetTimeCurrent)
      timeLastLog = timeCurrent;

    display.display();
    break;

  case STATE_SP_WAIT:
    displayFrame();
    display.setCursor(xOffset+(2*(5+1)), yOffset+(1*lineSpacing));
    display.print(F("SP "));
    display.print(currentSetpoint+1);
    display.display();
    if (menuButton == HIGH)
      currentState = STATE_SP;
    break;

  case STATE_SP:
    displayFrame();
    display.setCursor(xOffset+(2*(5+1)), yOffset+(1*lineSpacing));
    display.print(F("SP "));
    display.print(currentSetpoint+1);
    display.display();
    if (menuButton == LOW)
    {
      currentState = STATE_SP_WAIT;
      currentSetpoint++;
      if (currentSetpoint >= 4)
        currentState = STATE_CFG_WAIT;
    }
    else
    if (enterButton == LOW)
      currentState = STATE_SP_ENTER_WAIT;
    break;

  case STATE_SP_ENTER_WAIT:
    if (enterButton == HIGH)
    {
      currentState = STATE_SP_ENTER;
      SetpointNew = Setpoint[currentSetpoint];
    }
    break;

  case STATE_SP_ENTER:
    if (menuButton == LOW)
      currentState = STATE_SP_WAIT;
    else
    if (upButton == LOW)
    {
      SetpointNew += 0.1;
      buttonDownIncrementTime = 0;
      currentState = STATE_SP_ENTER_UP_WAIT;
    }
    else
    if (dnButton == LOW)
    {
      SetpointNew -= 0.1;
      buttonDownStartTime = millis() * DELAY_DIVISOR;
      buttonDownIncrementTime = 0;
      currentState = STATE_SP_ENTER_DN_WAIT;
    }
    else
    if (enterButton == LOW)
      currentState = STATE_SP_UPDATE_WAIT;
    displayFrame();
    display.setCursor(xOffset+(2*(5+1)), yOffset+(1*lineSpacing));
    display.print(F("SP "));
    display.print(currentSetpoint+1);
    display.setCursor(xOffset+(2*(5+1)), yOffset+(2*lineSpacing));
    display.print(SetpointNew);
    display.display();
    break;

  case STATE_SP_ENTER_UP_WAIT:
    if (upButton == HIGH)
      currentState = STATE_SP_ENTER;
    else
    {
      buttonDownCurrentTime = millis() * DELAY_DIVISOR;
      if (buttonDownCurrentTime > (buttonDownStartTime + BUTTON_HOLD_TIME))
      {
        if (buttonDownIncrementTime == 0 ||
            buttonDownCurrentTime > (buttonDownIncrementTime + BUTTON_HOLD_DELTA))
        {
          SetpointNew += 0.1;
          buttonDownIncrementTime = buttonDownCurrentTime;
        }
      }
    }
    displayFrame();
    display.setCursor(xOffset+(2*(5+1)), yOffset+(1*lineSpacing));
    display.print(F("SP "));
    display.print(currentSetpoint+1);
    display.setCursor(xOffset+(2*(5+1)), yOffset+(2*lineSpacing));
    display.print(SetpointNew);
    display.display();
    break;

  case STATE_SP_ENTER_DN_WAIT:
    if (dnButton == HIGH)
      currentState = STATE_SP_ENTER;
    else
    {
      buttonDownCurrentTime = millis() * DELAY_DIVISOR;
      if (buttonDownCurrentTime > (buttonDownStartTime + BUTTON_HOLD_TIME))
      {
        if (buttonDownIncrementTime == 0 ||
            buttonDownCurrentTime > (buttonDownIncrementTime + BUTTON_HOLD_DELTA))
        {
          SetpointNew -= 0.1;
          buttonDownIncrementTime = buttonDownCurrentTime;
        }
      }
    }
    displayFrame();
    display.setCursor(xOffset+(2*(5+1)), yOffset+(1*lineSpacing));
    display.print(F("SP "));
    display.print(currentSetpoint+1);
    display.setCursor(xOffset+(2*(5+1)), yOffset+(2*lineSpacing));
    display.print(SetpointNew);
    display.display();
    break;

  case STATE_SP_UPDATE_WAIT:
    displayFrame();
    display.setCursor(xOffset+(2*(5+1)), yOffset+(1*lineSpacing));
    display.print(F("SP "));
    display.print(currentSetpoint+1);
    display.setCursor(xOffset+(2*(5+1)), yOffset+(2*lineSpacing));
    display.print(F("StRd"));
    display.display();
    if (enterButton == HIGH)
    {
      Serial.println(F("Updating setpoint table."));
      Serial.print(F("currentSetpoint = ")); Serial.println(currentSetpoint);
      for (int iSetpoint=0; iSetpoint<240; iSetpoint++)
      {
        Setpoints_Thousandths[currentSetpoint][iSetpoint] = (unsigned int) (SetpointNew*1000.0);
      }

      Setpoint[currentSetpoint] = SetpointNew;
      currentState = STATE_SP;
      currentSetpoint++;
      if (currentSetpoint >= 4)
        currentState = STATE_CFG_WAIT;
    }
    break;


  case STATE_CFG_WAIT:
    displayFrame();
    display.setCursor(xOffset+(2*(5+1)), yOffset+(1*lineSpacing));     // Start at top-left
corner
    display.print(F("CNFG"));
    if (menuButton == HIGH)
      currentState = STATE_CFG;
    break;

  case STATE_CFG:
    display.setCursor(xOffset+(2*(5+1)), yOffset+(1*lineSpacing));     // Start at top-left
corner
    display.print(F("CNFG"));
    if (menuButton == LOW)
      currentState = STATE_RUN_WAIT;
    else
    if (enterButton == LOW)
      currentState = STATE_CFG_ENTER_WAIT;
    break;

  case STATE_RUN_WAIT:
    displayFrame();
    display.setCursor(xOffset+(2*(5+1)), yOffset+(1*lineSpacing));     // Start at top-left
corner
    display.print(F("RUN"));
    if (menuButton == HIGH)
    {
      currentState = STATE_RUN;
      displayRun();
      for (int i=0; i<maxRTD; i++)
        prevTemp[i] = 0.0;
    }
    break;

  default:
    break;
}

display.display();

}

void displayFrame()
{
display.clearDisplay();
for(int i=0; i<1; i++) {
display.drawRoundRect(i, i, display.width()-2*i, display.height()-2*i,
    display.height()/4, WHITE);
}
}

// Initialize the SD for operations
// If the LOGGING.CSV file is not present create the file with the first line of column headings
void SetupSDCardOperations()
{
displayFrame();
display.setCursor(xOffset, yOffset+(1*lineSpacing));
display.print(F("*** STATUS ***  "));
display.setCursor(xOffset, yOffset+(2*lineSpacing));
display.print(F("SD Init Start   "));
display.display();

Serial.println(F("SD.begin(chipSelectSDCard)"));

if (!SD.begin(chipSelectSDCard)) {
Serial.println(F("*** FAILED ***"));
  displayFrame();
  display.setCursor(xOffset, yOffset+(1*lineSpacing));
  display.print(F("*** ERROR ***   "));
  display.setCursor(xOffset, yOffset+(2*lineSpacing));
  display.print(F("SD Init Failed  "));
  display.setCursor(xOffset, yOffset+(3*lineSpacing));
  display.print(F("System HALTED!"));
  display.display();
  while (1);
}
Serial.println(F("*** SUCCESS ***"));

delay(2000/DELAY_DIVISOR);

displayFrame();
display.setCursor(xOffset, yOffset+(1*lineSpacing));
display.print(F("* Test CLOCK.CSV"));
display.setCursor(xOffset, yOffset+(2*lineSpacing));
if (SD.exists("CLOCK.CSV"))
{
  //display.print(F("  Set Clock     "));

  fileSDCard = SD.open("CLOCK.CSV");
  if (fileSDCard)
  {
    char *ptr3 = 0;

    if (fileSDCard.available())
    {
      char strClockSetting[256];
      char strClockSettingCopy[256];
      fileSDCard.read(strClockSetting, sizeof(strClockSetting));
      fileSDCard.close();
      strClockSetting[sizeof(strClockSetting)-1] = '\0';
      char *ptr1 = strchr(&strClockSetting[0],'\n');
      if (ptr1 != 0)
      {
          *ptr1++ = '\0';
      }
      Serial.println(F("Set Clock:"));
      Serial.println(strClockSetting);

      ptr3 = strchr(ptr1,'\n');
      if (ptr3 != 0)
      {
          *ptr3++ = '\0';
          strcpy(strClockSettingCopy, ptr1);
      }
      Serial.println(ptr1);

      int iDateTime[6] = {0,0,0,0,0,0};
      for (int i=0; i<6; i++)
      {
        char *ptr2 = strchr(ptr1,',');
        if (ptr2 != 0)
        {
          *ptr2 = '\0';
          iDateTime[i] = atoi(ptr1);
          ptr1 = &ptr2[1];
        }
        else
        {
          if (i == 5)
            iDateTime[5] = atoi(ptr1);
          break;
        }
      }


rtc.adjust(DateTime(iDateTime[0],iDateTime[1],iDateTime[2],iDateTime[3],iDateTime[4],iDateTime[5]))
;
```

```
        if(SD.exists("PRVCLOCK.CSV"))
        {
          SD.remove("PRVCLOCK.CSV");
        }
        SD.remove("CLOCK.CSV");

        fileSDCard = SD.open("PRVCLOCK.CSV", FILE_WRITE);
        if (fileSDCard != 0)
        {
          fileSDCard.println("\"Year\",\"Month\",\"Day\",\"Hour\",\"Minute\",\"Second\"");
          fileSDCard.println(strClockSettingCopy);
          fileSDCard.close();
        }

        //display.setCursor(xOffset, yOffset+(2*lineSpacing));
        display.print(F("* processed *"));
        display.display();
        delay(2000/DELAY_DIVISOR);
      }
      else
      {
        fileSDCard.close();
      }
    }

  }
  else
  {
    display.print(F("* does not exist"));
    display.display();
  }
  delay(2000/DELAY_DIVISOR);

// TEST.CSV Processing
  displayFrame();
  display.setCursor(xOffset, yOffset+(1*lineSpacing));
  display.print(F("* Test TEST.CSV"));
  display.setCursor(xOffset, yOffset+(2*lineSpacing));
  if (SD.exists("TEST.CSV"))
  {
    fileSDCard = SD.open("TEST.CSV");
    if (fileSDCard)
    {
      if (fileSDCard.available())
      {
        DoTests();

        displayFrame();
        display.setCursor(xOffset, yOffset+(1*lineSpacing));
        display.print(F("* TEST System"));
        display.setCursor(xOffset, yOffset+(3*lineSpacing));
        display.print(F("* processed *"));
        display.display();
        delay(2000/DELAY_DIVISOR);

        displayFrame();
        display.setCursor(xOffset, yOffset+(1*lineSpacing));
        display.print(F("* Test TEST.CSV"));
        display.setCursor(xOffset, yOffset+(3*lineSpacing));
        display.print(F("System HALTED!"));
        display.display();
        while (1);
      }
      else
      {
        fileSDCard.close();
      }
    }
  }
  else
  {
    display.print(F("* does not exist"));
    display.display();
  }
  delay(2000/DELAY_DIVISOR);

// CONTROL.CSV Processing
  displayFrame();
  display.setCursor(xOffset, yOffset+(1*lineSpacing));
  display.print(F("* Test CONTROL.CSV"));
  display.setCursor(xOffset, yOffset+(2*lineSpacing));
  if (SD.exists("CONTROL.CSV"))
  {
    fileSDCard = SD.open("CONTROL.CSV");
    if (fileSDCard)
    {
      char *ptr3 = 0;

      if (fileSDCard.available())
      {
        char strControlSetting[256];
        char strControlSettingCopy[256];
        fileSDCard.read(strControlSetting, sizeof(strControlSetting));
        fileSDCard.close();
        strControlSetting[sizeof(strControlSetting)-1] = '\0';
        char *ptr1 = strchr(&strControlSetting[0],'\n');
        if (ptr1 != 0)
        {
            *ptr1++ = '\0';
        }
        Serial.println(F("Set Control:"));
        Serial.println(strControlSetting);

        ptr3 = strchr(ptr1,'\n');
        if (ptr3 != 0)
        {
            *ptr3++ = '\0';
            strcpy(strControlSettingCopy, ptr1);
        }
        Serial.println(ptr1);

        int iControl[4] = {0,0,0,0};
        for (int i=0; i<4; i++)
        {
          char *ptr2 = strchr(ptr1,',');
          if (ptr2 != 0)
          {
            *ptr2 = '\0';
            iControl[i] = atoi(ptr1);
            ptr1 = &ptr2[1];
          }
          else
          {
            if (i == 3)
              iControl[3] = atoi(ptr1);
            break;
          }
        }
        Kp = iControl[0];
        Ki = iControl[1];
        Kd = iControl[2];
        POn = iControl[3];

        display.print(F("* processed *"));
        display.display();
        delay(2000/DELAY_DIVISOR);
      }
      else
      {
        fileSDCard.close();
      }
    }
  }
  else
  {
    display.print(F("* does not exist"));
    display.display();
  }
  delay(2000/DELAY_DIVISOR);
////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////


// OFFSETS.CSV Processing
  displayFrame();
  display.setCursor(xOffset, yOffset+(1*lineSpacing));
  display.print(F("* Test OFFSETS.CSV"));
  display.setCursor(xOffset, yOffset+(2*lineSpacing));
  if (SD.exists("OFFSETS.CSV"))
  {
    fileSDCard = SD.open("OFFSETS.CSV");
    if (fileSDCard)
    {
      char *ptr3 = 0;

      if (fileSDCard.available())
      {
        char strOffsets[256];
        char strOffsetsCopy[256];
        fileSDCard.read(strOffsets, sizeof(strOffsets));
        fileSDCard.close();
        strOffsets[sizeof(strOffsets)-1] = '\0';
        char *ptr1 = strchr(&strOffsets[0],'\n');
        if (ptr1 != 0)
        {
            *ptr1++ = '\0';
        }
        Serial.println(F("Set Offsets:"));
        Serial.println(strOffsets);

        ptr3 = strchr(ptr1,'\n');
        if (ptr3 != 0)
        {
            *ptr3++ = '\0';
            strcpy(strOffsetsCopy, ptr1);
        }
        Serial.println(ptr1);

        for (int i=0; i<4; i++)
        {
          offsetTemp[i] = 0.0;
          char *ptr2 = strchr(ptr1,',');
          if (ptr2 != 0)
          {
            *ptr2 = '\0';
            offsetTemp[i] = atof(ptr1);
            ptr1 = &ptr2[1];
          }
          else
          {
            if (i == 3)
              offsetTemp[3] = atof(ptr1);
            break;
          }
        }
        display.print(F("* processed *"));
        display.display();
        delay(2000/DELAY_DIVISOR);
      }
      else
      {
        fileSDCard.close();
      }
    }
  }
  else
  {
    display.print(F("* does not exist"));
    display.display();
  }
  delay(2000/DELAY_DIVISOR);
////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////
```

```
#if 1

// MINPCTS.CSV Processing
  displayFrame();
  display.setCursor(xOffset, yOffset+(1*lineSpacing));
  display.print(F("* Test MINPCTS.CSV"));
  display.setCursor(xOffset, yOffset+(2*lineSpacing));
  if (SD.exists("MINPCTS.CSV"))
  {
    fileSDCard = SD.open("MINPCTS.CSV");
    if (fileSDCard)
    {
      char *ptr3 = 0;

      if (fileSDCard.available())
      {
        char strVal[256];
        char strValCopy[256];
        fileSDCard.read(strVal, sizeof(strVal));
        fileSDCard.close();
        strVal[sizeof(strVal)-1] = '\0';
        char *ptr1 = strchr(&strVal[0],'\n');
        if (ptr1 != 0)
        {
            *ptr1++ = '\0';
        }
        Serial.println(F("Set Min Pcts:"));
        Serial.println(strVal);

        ptr3 = strchr(ptr1,'\n');
        if (ptr3 != 0)
        {
            *ptr3++ = '\0';
            strcpy(strValCopy, ptr1);
        }
        Serial.println(ptr1);

        for (int i=0; i<4; i++)
        {
          MinPctDutyCycle[i] = 0.0;
          char *ptr2 = strchr(ptr1,',');
          if (ptr2 != 0)
          {
            *ptr2 = '\0';
            //NOTE: Converting Pct into range 0-255
            MinPctDutyCycle[i] = (atof(ptr1)/100.0)*255.0;
            ptr1 = &ptr2[1];
          }
          else
          {
            if (i == 3)
              //NOTE: Converting Pct into range 0-255
              MinPctDutyCycle[3] = (atof(ptr1)/100.0)*255.0;
            break;
          }
        }
        display.print(F("* processed *"));
        display.display();
        delay(2000/DELAY_DIVISOR);
      }
      else
      {
        fileSDCard.close();
      }
    }
  }
  else
  {
    display.print(F("* does not exist"));
    display.display();
  }
  delay(2000/DELAY_DIVISOR);
////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////


// MAXPCTS.CSV Processing
  displayFrame();
  display.setCursor(xOffset, yOffset+(1*lineSpacing));
  display.print(F("* Test MAXPCTS.CSV"));
  display.setCursor(xOffset, yOffset+(2*lineSpacing));
  if (SD.exists("MAXPCTS.CSV"))
  {
    fileSDCard = SD.open("MAXPCTS.CSV");
    if (fileSDCard)
    {
      char *ptr3 = 0;

      if (fileSDCard.available())
      {
        char strVal[256];
        char strValCopy[256];
        fileSDCard.read(strVal, sizeof(strVal));
        fileSDCard.close();
        strVal[sizeof(strVal)-1] = '\0';
        char *ptr1 = strchr(&strVal[0],'\n');
        if (ptr1 != 0)
        {
            *ptr1++ = '\0';
        }
        Serial.println(F("Set Max Pcts:"));
        Serial.println(strVal);

        ptr3 = strchr(ptr1,'\n');
        if (ptr3 != 0)
        {
            *ptr3++ = '\0';
            strcpy(strValCopy, ptr1);
        }
        Serial.println(ptr1);

        for (int i=0; i<4; i++)
        {
          MaxPctDutyCycle[i] = 0.0;
          char *ptr2 = strchr(ptr1,',');
          if (ptr2 != 0)
          {
            *ptr2 = '\0';
            //NOTE: Converting Pct into range 0-255
            MaxPctDutyCycle[i] = (atof(ptr1)/100.0)*255.0;
            ptr1 = &ptr2[1];
          }
          else
          {
            if (i == 3)
              //NOTE: Converting Pct into range 0-255
              MaxPctDutyCycle[3] = (atof(ptr1)/100.0)*255.0;
            break;
          }
        }
        display.print(F("* processed *"));
        display.display();
        delay(2000/DELAY_DIVISOR);
      }
      else
      {
        fileSDCard.close();
      }
    }
  }
  else
  {
    display.print(F("* does not exist"));
    display.display();
  }
  delay(2000/DELAY_DIVISOR);
////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////
#endif


// SETPNTx.CSV Processing
  for (int iUnit=1; iUnit<=maxRTD; iUnit++)
  {
    char szSETPNTx[] = "SETPNTx.CSV";
    szSETPNTx[6] = '0'+iUnit;
    displayFrame();
    display.setCursor(xOffset, yOffset+(1*lineSpacing));
    display.print(F("* Test "));
    display.print(szSETPNTx);
    display.setCursor(xOffset, yOffset+(2*lineSpacing));
    if (SD.exists(szSETPNTx))
    {
      fileSDCard = SD.open(szSETPNTx);
      if (fileSDCard)
      {
        char *ptr3 = 0;
        if (fileSDCard.available())
        {
          ReadSetpoints(&Setpoints_Thousandths[iUnit-1][0]);

          now = rtc.now();
          int currentHour = now.hour();
          int currentMin  = now.minute();
          int index = (currentHour*10) + ((currentMin*10)/60);
          Setpoint[iUnit-1] = double(Setpoints_Thousandths[iUnit-1][index]) / 1000.0;

          display.print(F("* processed *"));
          display.display();
          delay(2000/DELAY_DIVISOR);
        }
        else
        {
          fileSDCard.close();
        }
      }
    }
    else
    {
      display.print(F("* does not exist"));
      display.display();
    }
    delay(2000/DELAY_DIVISOR);
  }
////////////////////////////////////////////////////////////////////////////////////////////////////
//////////////

  // LOGGING.CSV file processing
  fileSDCard = SD.open("LOGGING.CSV");
  if (fileSDCard)
  {
    if (fileSDCard.available())
    {
    }
    fileSDCard.close();
  }
  else
  {
    fileSDCard = SD.open("LOGGING.CSV", FILE_WRITE);
    if (fileSDCard)
    {

fileSDCard.println("\"Date\",\"Time\",\"Unit\",\"Setpt\",\"Temp\",\"Delta\",\"Offset\",\"Output\",\
"DutyCycle\",\"Dir\"");
      fileSDCard.close();
    }
    else
    {
      displayFrame();
      display.setCursor(xOffset, yOffset+(1*lineSpacing));
      display.print(F("*** ERROR ***   "));
      display.setCursor(xOffset, yOffset+(2*lineSpacing));
      display.print(F("SD Write Failed "));
      display.display();
      while (1);
    }
  }
  SD.end();

  displayFrame();
  display.setCursor(xOffset, yOffset+(1*lineSpacing));
  display.print(F("*** STATUS ***  "));
  display.setCursor(xOffset, yOffset+(2*lineSpacing));
  display.print(F("SD Init Finish  "));
  display.display();
  delay(2000/DELAY_DIVISOR);

  SDLogging("Start Up", 0.0, 0.0, 0.0, 0.0, 0.0, "");
}

void SDLogging(char *szUnit, double setpoint, double temp, double delta, double offset, double
output, char *strDir)
{
  if (!SD.begin(chipSelectSDCard))
  {
#if 0
    bSDLogFail = true;
    iToggle++;
    if ((iToggle & B00000001) == 0)
    {
      display.setCursor(xOffset, yOffset+(2*lineSpacing));
      display.print(F("SD LogFail"));
    }
#endif
    return;
  }
  bSDLogFail = false;
  iToggle = 0;

//  if ((strcmp((const char*) szUnit, "") != 0) || bForceOneMinuteLogging)
  {
    fileSDCard = SD.open("LOGGING.CSV", FILE_WRITE);

    // if the file opened okay, write to it:
    if (fileSDCard)
    {
      fileSDCard.print(now.year(), DEC);
      fileSDCard.print("/");
      fileSDCard.print(now.month(), DEC);
      fileSDCard.print("/");
      fileSDCard.print(now.day(), DEC);
      fileSDCard.print(",");
      fileSDCard.print(now.hour(), DEC);
      fileSDCard.print(":");
      fileSDCard.print(now.minute(), DEC);
      fileSDCard.print(":");
      fileSDCard.print(now.second(), DEC);
      fileSDCard.print(",");
      fileSDCard.print(szUnit);
      fileSDCard.print(",");
      fileSDCard.print(setpoint);
      fileSDCard.print(",");
      fileSDCard.print(temp);
      fileSDCard.print(",");
      fileSDCard.print(delta);
      fileSDCard.print(",");
      fileSDCard.print(offset);
      fileSDCard.print(",");
      fileSDCard.print(output);
      fileSDCard.print(",");
      double DutyCycle = (output/255.0)*100.0;
      fileSDCard.print(DutyCycle);
      fileSDCard.print("%,");
      fileSDCard.print(strDir);
      fileSDCard.println("");
      fileSDCard.close();
      SD.end();
    }
    else
    {
      // if the file didn't open, display an error:
      displayFrame();
      display.setCursor(xOffset, yOffset+(1*lineSpacing));
      display.print(F("*** ERROR ***   "));
      display.setCursor(xOffset, yOffset+(2*lineSpacing));
      display.print(F("Open LOGGING.CSV"));
      display.display();
      delay(2000/DELAY_DIVISOR);
    }
  }
}

void OledDisplayPrintTwoDigits(int iVal)
{
  if (iVal < 10)
    display.print(F("0"));
  display.print(iVal, DEC);
}

bool readLine(char* line, size_t maxLen) {
  for (size_t n = 0; n < maxLen; n++) {
    int c = fileSDCard.read();
    if ( c < 0 && n == 0) return false;  // EOF
    if (c < 0 || c == '\n') {
      line[n] = 0;
      return true;
    }
    line[n] = c;
  }
  return false; // line too long
}

bool readVals(int* iHour, int* iMinute, double* temp) {
  char line[40], *ptr, *str;
  if (!readLine(line, sizeof(line))) {
    return false;  // EOF or too long
  }
  ptr = &line[0];
  *iHour = atoi(ptr);
  while (*ptr) {
    if (*ptr++ == ':') break;
  }
  *iMinute = atoi(ptr);
  while (*ptr) {
    if (*ptr++ == ',') break;
  }
  *temp = strtod(ptr, &str);
  return str != ptr;  // true if number found
}

void ReadSetpoints(unsigned int* Setpoints_Thousandths)
{
  int iHour, iMinute;
  int index;
  double temp;
  long iThousandths;

  while (readVals(&iHour, &iMinute, &temp)) {
    iThousandths = (unsigned int) (temp*1000.0);
    index = (iHour*10) + ((iMinute*10)/60);
    Setpoints_Thousandths[index] = iThousandths;
#if 0
    Serial.print(F("Hr: "));
    Serial.print(iHour);
    Serial.print(F(" Mn: "));
    Serial.print(iMinute);
    Serial.print(F(" Temp: "));
    Serial.print(temp,3);
    Serial.print(F(" iThousandths: "));
    Serial.print(iThousandths);
    Serial.print(F(" Index: "));
    Serial.println(index);
#endif
  }
}

bool readTest(int* iUnit, char* cHeatOrCool, int* iSeconds) {
  char line[40], *ptr, *str;
  if (!readLine(line, sizeof(line))) {
    return false;  // EOF or too long
  }
  ptr = &line[0];
  *iUnit = atoi(ptr);
  *cHeatOrCool = ptr[1];

  while (*ptr) {
    if (*ptr++ == ',') break;
  }
  *iSeconds = atoi(ptr);
  return str != ptr;  // true if number found
}
```

```
void DoTests()
{
  int iUnit;
  char cHeatOrCool;
  int iSeconds;

  for (int i=0; i< 4; i++)
  {
    pinMode(HeaterUnits[i], OUTPUT);
    analogWrite(HeaterUnits[i], 0.0);
    pinMode(CoolerUnits[i], OUTPUT);
    analogWrite(CoolerUnits[i], 0.0);
  }

  while (readTest(&iUnit, &cHeatOrCool, &iSeconds)) {
    Serial.print(F("Unit: "));
    Serial.print(iUnit);
    Serial.print(cHeatOrCool);
    Serial.print(F(" Seconds: "));
    Serial.println(iSeconds);

    displayFrame();
    display.setCursor(xOffset, yOffset+(1*lineSpacing));
    display.print(F("* TEST System"));
    display.setCursor(xOffset, yOffset+(2*lineSpacing));
    display.print(F("Unit: "));
    display.print(iUnit);
    display.print(cHeatOrCool);
    display.setCursor(xOffset, yOffset+(3*lineSpacing));
    display.print(F("Secs: "));
    display.print(iSeconds);
    display.display();

    if (iUnit >= 1 && iUnit <= 4)
    {
      if (cHeatOrCool == 'H')
        analogWrite(HeaterUnits[iUnit-1], 255.0);
      else
      if (cHeatOrCool == 'C')
        analogWrite(CoolerUnits[iUnit-1], 255.0);
      else
      {
        break;
      }

      delay((1000*iSeconds)/DELAY_DIVISOR);

      if (cHeatOrCool == 'H')
        analogWrite(HeaterUnits[iUnit-1], 0.0);
      else
      if (cHeatOrCool == 'C')
        analogWrite(CoolerUnits[iUnit-1], 0.0);
    }
  }
}
```

```
/* Converts a hex character to its integer value */
char from_hex(char ch) {
  return isdigit(ch) ? ch - '0' : tolower(ch) - 'a' + 10;
}

/* Converts an integer value to its hex character*/
char to_hex(char code) {
  static char hex[] = "0123456789abcdef";
  return hex[code & 15];
}

#if 0
void url_encode(char *buf, char *str) {
  char *pstr = str, *pbuf = buf;
  while (*pstr) {
    if (isalnum(*pstr) || *pstr == '-' || *pstr == '_' || *pstr == '.' || *pstr == '~')
      *pbuf++ = *pstr;
    else if (*pstr == ' ')
      *pbuf++ = '+';
    else
      *pbuf++ = '%', *pbuf++ = to_hex(*pstr >> 4), *pbuf++ = to_hex(*pstr & 15);
    pstr++;
  }
  *pbuf = '\0';
}
#endif

void url_decode(char *buf, char *str) {
  char *pstr = str, *pbuf = buf;
  while (*pstr) {
    if (*pstr == '%') {
      if (pstr[1] && pstr[2]) {
        *pbuf++ = from_hex(pstr[1]) << 4 | from_hex(pstr[2]);
        pstr += 2;
      }
    } else if (*pstr == '+') {
      *pbuf++ = ' ';
    } else {
      *pbuf++ = *pstr;
    }
    pstr++;
  }
  *pbuf = '\0';
}

bool readLineFileTransfer(char* line, size_t maxLen)
{
  char sBuffer[128];
  char sTmp[128];
//  char sBuffer[256];
//  char sTmp[256];

  for (size_t n = 0; n < maxLen; n++)
  {
    int c = fileSDCard.read();
    if ( c < 0 && n == 0) return false;  // EOF
    if (c < 0 || c == '\n')
    {
      sBuffer[n] = '\n';
      sBuffer[n+1] = 0;
      if (sBuffer[13] != ':' ||
          sBuffer[11] == ':' ||
          sBuffer[12] == ':')
      {
        if (sBuffer[7] != '/')
        {
          strcpy(sTmp, &sBuffer[5]);
          strcpy(&sBuffer[6], sTmp);
          sBuffer[5] = '0';
        }
        if (sBuffer[10] != ',')
        {
          strcpy(sTmp, &sBuffer[8]);
          strcpy(&sBuffer[9], sTmp);
          sBuffer[8] = '0';
        }
        if (sBuffer[13] != ':')
        {
          strcpy(sTmp, &sBuffer[11]);
          strcpy(&sBuffer[12], sTmp);
          sBuffer[11] = '0';
        }
      }
      strcpy(line, sBuffer);
      return true;
    }
    sBuffer[n] = c;
  }
  return false; // line too long
}


bool seekNextLineStart(size_t maxLen) {
  for (size_t n = 0; n < maxLen; n++) {
    int c = fileSDCard.read();
    if ( c < 0 && n == 0) return false;  // EOF
    if (c < 0 || c == '\n') {
      return true;
    }
  }
  return false; // line too long
}

void FileTransfer(void)
{
  if (Serial1.available() > 0)
  {
#if 1
//char *cEncodedBuffer = malloc(256);
char cEncodedBuffer[1024 + 64];
char *cDecodedBuffer = cEncodedBuffer;
```

```
//char cDecodedBuffer[256 + 64];
char *pLogging = &cEncodedBuffer[0];

char sFilename[13] = "";
char sFilenameBak[13] = "";

#endif
    int iLen;
    iLen = Serial1.readBytesUntil('\r', cEncodedBuffer, sizeof(cEncodedBuffer) - 1);
    cEncodedBuffer[iLen] = '\0';
#if 1
    Serial.print(F("Input len = "));
    Serial.println(iLen);
    Serial.print(F("["));
    char cSave = cEncodedBuffer[40];
    cEncodedBuffer[40] = '\0';
    Serial.print(cEncodedBuffer);
    cEncodedBuffer[40] = cSave;
    Serial.println(F("]"));
#endif

    char *ptr;
#if 1
    ptr = strstr(cEncodedBuffer, "?BOF");
    if (ptr != 0)
    {
      Serial.println(F("BOF"));
      Serial1.print("1");
      bFileUploading = true;
      return;
    }

    ptr = strstr(cEncodedBuffer, "?EOF");
    if (ptr != 0)
    {
      Serial.println(F("EOF"));
      Serial1.print("1");
      bFileUploading = false;
      return;
    }

    ptr = strstr(cEncodedBuffer, "?RESET");
    if (ptr != 0)
    {
      Serial.println(F("RESET"));
      Serial1.print("1");
      resetFunc();
      return;
    }
#endif

    ptr = strstr(cEncodedBuffer, "&d=");
    if (ptr == 0)
      ptr = strstr(cEncodedBuffer, "?h=");
    if (ptr == 0)
    {
```

```
#if DEBUGGING
      Serial.println(F("INVALID BUFFER"));
#endif
      return;
    }

    //cDecodedBuffer[0] = '\0';

    char *pFilename = strstr(cEncodedBuffer, "GET /?f=");
    if (pFilename != 0)
    {
Serial.println(F("SD.begin(chipSelectSDCard)"));

  if (!SD.begin(chipSelectSDCard)) {
Serial.println(F("*** FAILED ***"));
    displayFrame();
    display.setCursor(xOffset, yOffset+(1*lineSpacing));
    display.print(F("*** ERROR ***   "));
    display.setCursor(xOffset, yOffset+(2*lineSpacing));
    display.print(F("SD Init Failed  "));
    display.setCursor(xOffset, yOffset+(3*lineSpacing));
    display.print(F("System HALTED!"));
    display.display();
    while (1);
  }
Serial.println(F("*** SUCCESS ***"));

      char *pOffset = strstr(pFilename, "&o=");
      if (pOffset != 0)
      {
        char *pSize = strstr(pOffset, "&s=");
        if (pSize != 0)
        {
          char *pData = strstr(pSize, "&d=");
          if (pData != 0)
          {
            // Eureka !
            //bFileUploading = true;

            pOffset[0] = '\0';
            pSize[0] = '\0';
            int iOffset = atoi(&pOffset[3]);
            pData[0] = '\0';
            int iSize = atoi(&pSize[3]);

            iLen = strlen(&pData[3]);

            strcpy(sFilename, &pFilename[8]);
            strcpy(sFilenameBak, &pFilename[8]);
            char *pFiletype = strstr(sFilenameBak, ".");
            if (pFiletype != 0)
              strcpy(pFiletype, ".BAK");

#if DEBUGGING
            Serial.print(F("Filename: ["));
            Serial.print(sFilename);
            Serial.println(F("]"));
            Serial.print(F("Offset: "));
            Serial.println(iOffset);
            Serial.print(F("Size: "));
            Serial.println(iSize);
            Serial.print(F("Encoded len = "));
            Serial.println(iLen);
#endif

            url_decode(cDecodedBuffer, &pData[3]);

            iLen = strlen(cDecodedBuffer);
#if 1
            Serial.print(F("Decoded len = "));
            Serial.println(iLen);
#endif
#if DEBUGGING
            Serial.print(F("["));
            char cSave = cDecodedBuffer[40];
            cDecodedBuffer[40] = '\0';
            Serial.print(cDecodedBuffer);
            cDecodedBuffer[40] = cSave;
            Serial.println(F("]"));
            //Serial.println(F("Data: "));
            //Serial.println(cDecodedBuffer);
#endif

            if (iOffset == 0)
            {
#if DEBUGGING
              Serial.print(F("sFilenameBak: "));
              Serial.println(sFilenameBak);
#endif
              if (SD.exists(sFilenameBak))
              {
#if DEBUGGING
                Serial.print(F("SD.remove(sFilenameBak) = "));
#endif
                int iRet = SD.remove(sFilenameBak);
#if DEBUGGING
                Serial.println(iRet);
#endif
              }

#if DEBUGGING
              Serial.print(F("SD.open "));
              Serial.println(sFilename);
#endif

              fileSDCard = SD.open(sFilename);
              if (fileSDCard)
              {
                iLen = fileSDCard.size();
#if DEBUGGING
                Serial.print(F("iLen to copy: "));
                Serial.println(iLen);
                //Serial.print(F("freeMemory()="));
                //Serial.println(freeMemory());
#endif
                char *buffer = malloc(iLen);
                if (buffer == 0)
                {
#if DEBUGGING

                  Serial.println(F("MALLOC FAILED!"));
#endif
                }
                else
                {
#if DEBUGGING
                  Serial.println(F("fileSDCard.read"));
#endif
                  fileSDCard.read(buffer, iLen);
                  fileSDCard.close();

                  fileSDCard = SD.open(sFilenameBak, FILE_WRITE);
                  if (fileSDCard)
                  {
#if DEBUGGING
                    Serial.println(F("fileSDCard.write"));
#endif
                    fileSDCard.write(buffer, iLen);
                    fileSDCard.close();
                  }
                  free(buffer);
                }
#if DEBUGGING
                Serial.print(F("SD.remove(sFilename) = "));
#endif
                int iRet = SD.remove(sFilename);
#if DEBUGGING
                Serial.println(iRet);
#endif
              }
            }

#if DEBUGGING
            Serial.print(F("SD.open FILE_WRITE "));
            Serial.println(sFilename);
#endif
            fileSDCard = SD.open(sFilename, FILE_WRITE);
            if (fileSDCard)
            {
              iLen = strlen(cDecodedBuffer);
#if DEBUGGING
              Serial.print(F("fileSDCard.write "));
              Serial.println(sFilename);
              Serial.print(F("buffer: "));
              char cSave = cDecodedBuffer[40];
              cDecodedBuffer[40] = '\0';
              Serial.print(cDecodedBuffer);
              cDecodedBuffer[40] = cSave;
              Serial.print(F("len: "));
              Serial.println(iLen);
#endif
              fileSDCard.write(cDecodedBuffer, iLen);
              fileSDCard.close();
            }
          }
        }
      }
      SD.end();
    }
    else
    {
#if DEBUGGING
    Serial.print(F("["));
    Serial.print(cEncodedBuffer);
    Serial.println(F("]"));
#endif
      char *pHour = strstr(cEncodedBuffer, "GET /?h=");
      if (pHour == 0)
      {
        Serial1.print("0");
        return;
      }
      url_decode(cDecodedBuffer, &pHour[8]);
      pHour = &cDecodedBuffer[0];
      char *pColon = strstr(cDecodedBuffer, ":");
      if (pColon == 0)
      {
        Serial1.print("0");
        return;
      }
      pColon[1] = '\0';

      // we have hour for logging scan
      char sDateHour[32];
      char sTmp[32];
      strcpy(sDateHour, pHour);

#if DEBUGGING
      Serial.print(F("Hour = ["));
      Serial.print(sDateHour);
      Serial.println(F("]"));
#endif
      int iHourLen = strlen(sDateHour);

/*
xxxx/xx/xx,xx:
01234567890123
          1
*/
      if (strlen(sDateHour) != 14)
      {
        if (sDateHour[7] != '/')
        {
          strcpy(sTmp, &sDateHour[5]);
          strcpy(&sDateHour[6], sTmp);
          sDateHour[5] = '0';
        }
        if (sDateHour[10] != ',')
        {
          strcpy(sTmp, &sDateHour[8]);
          strcpy(&sDateHour[9], sTmp);
          sDateHour[8] = '0';
        }
        if (sDateHour[13] != ':')
        {
          sDateHour[12] = sDateHour[11];
          sDateHour[11] = '0';
          sDateHour[13] = ':';
        }
      }
      sDateHour[14] = '\0';

//      if (sDateHour[9] == '8')
//        sDateHour[9] = '7';

      iHourLen = strlen(sDateHour);
#if DEBUGGING
      Serial.print(F("Hour = ["));
      Serial.print(sDateHour);
      Serial.println(F("]"));
      Serial.print(F("iHourLen = "));
      Serial.println(iHourLen);
#endif

      long lFirst, lLast, lMiddle;
      int iRet;

      // open logging.csv and match hour
#if DEBUGGING
      Serial.println(F("SD.open LOGGING.CSV"));
#endif
      fileSDCard = SD.open("LOGGING.CSV", FILE_READ);
      if (fileSDCard)
      {
        long lFileSize = fileSDCard.size();
#if DEBUGGING
        Serial.print(F("Size = "));
        Serial.println(lFileSize);
#endif

        // Binary search to find date/time start
        lFirst = 0;
        lLast = lFileSize - 1;
        lMiddle = (lFirst+lLast)/2;
        while (lFirst <= lLast)
        {
#if DEBUGGING
          Serial.print(F("Seek = "));
          Serial.println(lMiddle);
#endif
```

```
          fileSDCard.seek(lMiddle);
          seekNextLineStart(256);
          if (readLineFileTransfer(pLogging, 256) == false)
            break;
#if DEBUGGING
          Serial.println(pLogging);
#endif
          char cSave = pLogging[iHourLen];
          pLogging[iHourLen] = '\0';
          iRet = strcmp(pLogging, sDateHour);
#if DEBUGGING
          Serial.println(pLogging);
          Serial.println(iRet);
#endif
          if (iRet < 0)
            lFirst = lMiddle+1;
          else
          if (iRet == 0)
          {
            pLogging[iHourLen] = cSave;
#if DEBUGGING
            Serial.println(pLogging);
#endif
            break;
          }
          else
            lLast = lMiddle - 1;
          lMiddle = (lFirst + lLast)/2;
         }

        //fileSDCard.seek(0);

//      if (sDateHour[9] == '7')
//        sDateHour[9] = '8';

        if (lMiddle < 16000)
            lMiddle = 0;
        else
            lMiddle -= 16000;

#if DEBUGGING
        Serial.print(F("lMiddle = "));
        Serial.println(lMiddle);
#endif
        fileSDCard.seek(lMiddle);


        int iOK = true;
        while (iOK)
        {
          if (readLineFileTransfer(pLogging, 256) == false)
            break;
          if (strncmp(pLogging, sDateHour, iHourLen) == 0)
          {
            while (iOK)
            {
              //Serial.print(pLogging);

              Serial1.print(pLogging);
              while (Serial1.available() == 0)
                ;
              Serial1.readBytesUntil('\r', cEncodedBuffer, sizeof(cEncodedBuffer) - 1);

              if (readLineFileTransfer(pLogging, 256) == false ||
                  strncmp(pLogging, sDateHour, iHourLen) != 0)
              {
                iOK = false;
                break;
              }
            }
          }
        }
#if DEBUGGING
        Serial.print(F("\n"));
#endif
      }
      else
      {
        Serial1.print("0");
        return;
      }
    }

    Serial1.print("1");

  }

}
```

ESP8266 WiFi Access Point INO Source Code

```
/*************************************************************************
  Access Point Web Server

  Original Code:  2021-09-08

  Tom Rolander, MSEE
  Mentor, Circuit Design & Software
  Miller Library, Fabrication Lab
  Hopkins Marine Station, Stanford University,
  120 Ocean View Blvd, Pacific Grove, CA 93950
  +1 831.915.9526 | rolander@stanford.edu

 *************************************************************************/

#define VERSION "Ver 0.3 2021-10-11"

#include <ESP8266WiFi.h>
#include <SoftwareSerial.h>

//////////////////////
// WiFi Definitions //
//////////////////////
const char WiFiAPPSK[] = "HMS-TOMPORT";

//IPAddress local_IP(192,168,4,22);
IPAddress local_IP_TOMPORT01(192,168,4,1);
IPAddress local_IP_TOMPORT02(192,168,4,2);

IPAddress gateway(192,168,4,9);
IPAddress subnet(255,255,255,0);

SoftwareSerial espSerial(4,15);


/////////////////////
// Pin Definitions //
/////////////////////
const int LED_PIN = 5; // Thing's onboard, green LED

WiFiServer server(80);

void setup()
{
  espSerial.begin(19200);
  Serial.begin(9600);
  delay(2500);
  Serial.printf("TOMPORT Access Point Web Server\n");
  Serial.printf(VERSION);
  Serial.printf("\r\n\r\n");
  delay(2500);
  pinMode(LED_PIN, OUTPUT);
  digitalWrite(LED_PIN, HIGH);

  setupWiFi();
  server.begin();
}

void loop()
{
  char in = 0;
  char line[512];
  int iLen;

  if (espSerial.available() > 0)
  {
#if 1
    //Serial.write(espSerial.read());

    iLen = espSerial.readBytesUntil('\n', line, sizeof(line)-1);
    line[iLen] = '\n';
    line[iLen+1] = '\0';
    Serial.write(line);

#else
    in = espSerial.read();
    line[0] = in;
    line[1] = 0;
    Serial.printf(&in);
#endif
  }

  // Check if a client has connected
  WiFiClient client = server.available();
  if (!client) {
    return;
  }

  // Read the first line of the request
//  String req = client.readStringUntil('\r');
//  Serial.println(req);

  char cReq[2048];
  int iLength = client.readBytesUntil('\r', cReq, sizeof(cReq)-1);
  cReq[iLength] = '\0';

  client.flush();

  // Match the request
  int val = -1; // We'll use 'val' to keep track of both the
                // request type (read/set) and value if set.
  Serial.println(cReq);
  if (strstr(cReq,"/led/0") != 0)
    val = 1; // Will write LED high
  else if (strstr(cReq,"/led/1") != 0)
    val = 0; // Will write LED low
  else if (strstr(cReq,"/?f=") != 0)
    val = -3; // send file encoded in the url
```

```
else if (strstr(cReq,"/?h=") != 0)
  val = -4; // get logging.csv for hour at xx/xx/xx,xx:
else if (strstr(cReq,"/?EOF") != 0)
  val = -5; // signal End Of File
else if (strstr(cReq,"/?BOF") != 0)
  val = -6; // signal Beg of File
else if (strstr(cReq,"/?RESET") != 0)
  val = -7; // reset the TOMPORT
// Otherwise request will be invalid. We'll say as much in HTML

// Set GPIO5 according to the request
if (val >= 0)
  digitalWrite(LED_PIN, val);

client.flush();

// Prepare the response. Start with the common header:
String s = "HTTP/1.1 200 OK\r\n";
s += "Content-Type: text/html\r\n\r\n";
if (val > -2)
  s += "<!DOCTYPE HTML>\r\n<html>\r\n";

// If we're setting the LED, print out a message saying we did
if (val >= 0)
{
  s += "LED is now ";
  s += (val)?"off":"on";
}
else if (val == -3 ||
         val == -5 ||
         val == -6 ||
         val == -7)
{
  char *str = strstr(cReq, " HTTP/");
  if (str != 0)
    *str = '\0';

  espSerial.print(cReq);
  espSerial.print('\r');

  while (espSerial.available() <= 0)
    ;
  int incomingByte = espSerial.read();
  Serial.print("Ret = ");
  Serial.println(incomingByte);

  s += "Line sent";
}
else if (val == -4)
{
  char *str = strstr(cReq, " HTTP/");
  if (str != 0)
    *str = '\0';
  espSerial.print(cReq);
  espSerial.print('\r');
//    s += "Date,Time,Unit,Setpt,Temp,Delta,Offset,Output,DutyCycle,Dir\r\n";

    int iOK = true;
    while (iOK)
    {
      if (espSerial.available() > 0)
      {
        iLen = espSerial.readBytesUntil('\n', line, sizeof(line)-1);
        if (iLen == 1)
          break;
        line[iLen] = '\n';
        line[iLen+1] = '\0';
//        Serial.print(line);
        s += String(line);
///        if (iLen == 1)
///          break;
        espSerial.print('\r');
      }
    }
  }
  else
  {
    s += "Invalid Request.<br> Try /led/1, or /led/0.";
  }

  if (val > -2)
    s += "</html>\n";

  // Send the response to the client
  client.print(s);
  delay(1);
  Serial.println("Client disonnected");

  // The client will actually be disconnected
  // when the function returns and 'client' object is detroyed
}

void setupWiFi()
{
  WiFi.mode(WIFI_AP);

  uint8_t mac[WL_MAC_ADDR_LENGTH];
  WiFi.softAPmacAddress(mac);
  String macID = String(mac[WL_MAC_ADDR_LENGTH - 2], HEX) +
                 String(mac[WL_MAC_ADDR_LENGTH - 1], HEX);
  macID.toUpperCase();
//  String AP_NameString = "TomPort-" + macID;

  String AP_NameString = "TOMPORT01";
  if (macID == "488B")
    AP_NameString = "TOMPORT02";

  char AP_NameChar[AP_NameString.length() + 1];
  memset(AP_NameChar, 0, AP_NameString.length() + 1);

  for (int i=0; i<AP_NameString.length(); i++)
```

```
    AP_NameChar[i] = AP_NameString.charAt(i);

  Serial.print("Setting soft-AP configuration ... ");
  Serial.println(WiFi.softAPConfig((macID == "488B") ? local_IP_TOMPORT02 : local_IP_TOMPORT01,
gateway, subnet) ? "Ready" : "Failed!");
  Serial.print('\n');

  WiFi.softAP(AP_NameChar, WiFiAPPSK);
}
```

Python TomPort Utilities

UploadFile.py

```
"""
/***************************************************************************
  Upload File (CSV)

  Original Code:  2021-10-11

  Tom Rolander, MSEE
  Mentor, Circuit Design & Software
  Miller Library, Fabrication Lab
  Hopkins Marine Station, Stanford University,
  120 Ocean View Blvd, Pacific Grove, CA 93950
  +1 831.915.9526 | rolander@stanford.edu

 ***************************************************************************/
"""

import requests
import sys
import urllib.parse

# total arguments
n = len(sys.argv)
if n != 3:
    print("Usage: python UploadFile.py TomPort filename")
    quit()

# Arguments passed
print("\nName of Python script:", sys.argv[0])

print("\nArguments passed: ")
print("TomPort [", sys.argv[1], "]")
print("UploadFile [", sys.argv[2], "]")

file = open(sys.argv[2],"rb")
mybytearray = bytearray()
byte = file.read(1)
while byte:
    mybytearray += byte
    byte = file.read(1)
file.close()

#print (mybytearray)

str1 = mybytearray.decode()
#print(str1)

str2 = urllib.parse.quote(str1)

print("len = " + str(len(str2)))
#print(str2)
```

```
if (sys.argv[1] == '1'):
    ipaddress = 'http://192.168.4.1'
else:
    ipaddress = 'http://192.168.4.2'

# stop TomPort scanning and prepare TomPort to receive file
url = ipaddress + '/?BOF';
print (url)
x = requests.get(url, timeout=30)
print(x.status_code)

# send file to TomPort, buffsize bytes at a chunk
buffsize = 1024
beg = 0
nxt = buffsize
end = len(str2)
if nxt > end:
    nxt = end

while beg < end:
    print("beg = " + str(beg))
    print("nxt = " + str(nxt))
    if str2[nxt-1] == '%':
        nxt = nxt - 1
    else:
        if str2[nxt-2] == '%':
            nxt = nxt - 2
    str3 = str2[beg:nxt]
    iLen  = len(str3)
    print("len = " + str(iLen))
#    print (str3)
    url = ipaddress + '/?f=' + sys.argv[2] + '&o=' + str(beg) + '&s=' + str(iLen) + '&d=' + str3;
    print (url)
    x = requests.get(url, timeout=30)
    print(x.status_code)
    beg = nxt
    nxt = beg + buffsize
    if nxt > end:
        nxt = end

# continue TomPort scanning after receiving file
url = ipaddress + '/?EOF';
print (url)
x = requests.get(url, timeout=30)
print(x.status_code)
```

DownloadFile.py

```
"""
/***************************************************************************
  Download File (CSV)

  Original Code:  2021-10-03

  Tom Rolander, MSEE
  Mentor, Circuit Design & Software
  Miller Library, Fabrication Lab
  Hopkins Marine Station, Stanford University,
  120 Ocean View Blvd, Pacific Grove, CA 93950
  +1 831.915.9526 | rolander@stanford.edu

 ***************************************************************************/
"""

import requests
import sys
import urllib.parse

# total arguments
n = len(sys.argv)
if n != 4:
    print("Usage: python DownloadFile.py TomPort Filename xx/xx/xx,xx:")
    quit()

print("\nArguments passed: ")
print("TomPort [", sys.argv[1], "]")
print("DownloadFile [", sys.argv[2], "]")
print("xx/xx/xx,xx: [", sys.argv[3], "]")

if (sys.argv[1] == '1'):
    ipaddress = 'http://192.168.4.1'
else:
    ipaddress = 'http://192.168.4.2'

url = ipaddress +  '/?h=' + urllib.parse.quote(sys.argv[3])
print (url)
x = requests.get(url, timeout=60)
print(x.status_code)
print(x.content.decode('ascii'))
open(sys.argv[2], 'wb').write(x.content)
```

ResetTomPort.py

```
"""
/************************************************************************
  Reset TomPort

  Original Code:  2021-10-11

  Tom Rolander, MSEE
  Mentor, Circuit Design & Software
  Miller Library, Fabrication Lab
  Hopkins Marine Station, Stanford University,
  120 Ocean View Blvd, Pacific Grove, CA 93950
  +1 831.915.9526 | rolander@stanford.edu

 ************************************************************************/
"""

import requests
import sys

# total arguments
n = len(sys.argv)
if n != 2:
    print("Usage: python ResetTomPort.py TomPort")
    quit()

print("\nArguments passed: ")
print("TomPort [", sys.argv[1], "]")

if (sys.argv[1] == '1'):
    ipaddress = 'http://192.168.4.1'
else:
    ipaddress = 'http://192.168.4.2'

url = ipaddress +  '/?RESET'
print (url)
x = requests.get(url, timeout=60)
print(x.status_code)
print(x.content.decode('ascii'))
```