SDAS – Student Data Acquisition System
Heating and Cooling Control of an Aluminum Block

Matt Smith

San Jose State University
Department of Mechanical and Aerospace Engineering
ME 106
Dr. Buford Furman

12/9/99

Abstract

This project encompasses the construction of a low-cost data acquisition system and the development of a measurement application using this data acquisition system. The purpose of the project is educational in nature and covers the following objectives:

- Application of physical and engineering principals
- Interfacing electromechanical systems to microcontrollers
- Analog-to-Digital and Digital-to-Analog concepts
- Program writing to perform applications
- Fundamental electronics

The system chosen was the SDAS (Student Data Acquisition System) and the application a simple heating and cooling device involving an Aluminum block. The system including the SDAS, the heating and cooling device, application program and supporting circuitry was constructed and tested successfully.

Acknowledgments

I would like to thank the following:

- Dr. Chris Braun – for numerous cross country e-mail and telephone technical support sessions
- Dr. Buff Furman – for endless help in troubleshooting and encouragement (not to mention his SDAS board!)
- Julia Howlett – for putting up with me and the messiest work area ever
- Symyx Technologies Inc.– for continuing support and understanding of my education as well as hardware and tools
- Tear Franks at Calculated Machining of Santa Clara, CA for the donation of the aluminum block
- Fellow students and friends for ideas and support

Table of Contents

Nomenclature

SDAS = Student Data Acquisition System
A2D = Analog to Digital
D2A = Digital to Analog
q" = heat flux (W/cm^2)
k = conductive heat transfer coefficient
q = heat rate (W)
T = temperature
i = current (A)
V = volts
DA0-3 = digital to analog channels
AD0-7 = analog to digital channels
DI0-0 = digital input channels
DO0-7 = digital output channels
PLD = programmable logic device
HAC = heating and cooling device

Executive Summary

The Problem:

The objective of this project was to come up with a data acquisition system and provide a measurement system to interact with it. This type of system needed to be operational in a given time period and display fundamental engineering principles and newly acquired skills in Mechatronics.

The Solution:

The solution to the objective was to construct a crude temperature control system. The SDAS was chosen for data acquisition and a heating and cooling device (HAC) consisting of an Aluminum block, a cartridge heater, thermoelectric, heat sink, temperature sensor and fan was constructed. In addition computer code in C and QBASIC was written and modified to control the system. The project took on three distinctive phases:

- System design
- SDAS and HAC construction
- System testing and coding

The system design was based on a sequence of steps that would occur within the system. Essentially, the heater would turned on and off through the digital output of the SDAS (and a MOSFET and relay circuit). This on/off switching will be controlled by a computer program that takes analog data in from the temperature sensor (LM35) in continuous manner and converting it to a digital value. When the temperature hits a preset upper limit, the power to the heater will be shut off and the cooling system turned on. Feedback can continue to flow and when the temperature hits a preset lower limit, the power can be turned back on to the heater and the process will continue to loop in this manner. A schematic of this process is shown in Figure1.

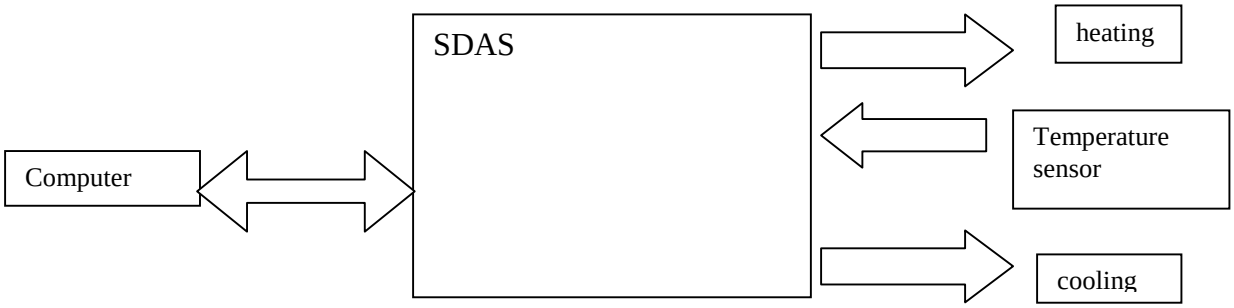

Figure 1 – ME 106 project concept – heater control

The construction of the individual components occurred at my home with some problems. The SDAS constructed by myself has no digital output from the PLD. Troubleshooting on this problem is forthcoming. An SDAS provided by Dr. Furman was used in the testing and operational phases.

The testing of the SDAS was achieved through test programs available at the SDAS web site (8). The main portion of the code that actually ran the system (however crudely) was written in QBASIC and is as follows:

```
WHILE (1)

dig# = A2D(0)
IF dig# < .605 THEN DigOut (&H1)
PRINT "Analog output is "; dig#
IF NOT dig# < .580 THEN DigOut (&H6)

WEND
END
```

Introduction

The SDAS board was chosen for this project for it's relatively low cost, robustness and functionality. The cost of the system is $100.00 and was ordered via e-mail and received in the mail. Some of the performance specifications of the SDAS are listed here:

- 8 channels, 12 bit analog input
- 4 channels, 8 bit analog output
- 8 bits digital input
- 8 bits digital output
- Sampling rates up to 40 kHz (analog in) to 300kHz (digital)
- Parallel port connection to computer
- Modular wire connector
- Software libraries

The system comes from the Colorado School of Mines and was put together by Dr. Christopher G. Braun. The idea for a low-cost DAQ system came about to try and change the way the school ran their lab courses. The idea was to "be an expansion of the laboratory experience outside of the laboratory that will help the students make connections between their education and their professional work." In this same spirit the SDAS was used in this project. Figures 2 and 3 show the block diagram and physical layout of the SDAS board.

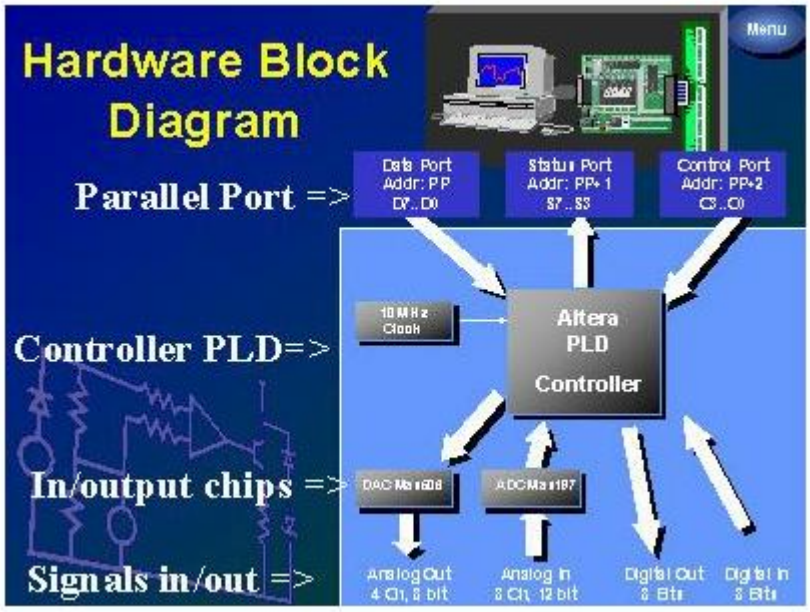

Figure 2 – Hardware block diagram for the SDAS
(Courtesy of Colorado School of Mines)

Some more specific specifications and qualities of the components of the SDAS are listed here:

- PLD
  - Altera EPM 7096 PLD, PLCC 68 package 96 macrocells, 5000 NAND gates

- A2D
  - Maxim Max 197, 12 bit successive approximation, 100kps, 1.22 mV to 4.88 mV resolution (depending on full scale voltage)
- D2A
  - Maxim Max 506, 4 Channel output (0-5 V), 19.53 mV/step resolution
- Clock
  - 10 MHz
- Power source
  - 9VAC wall transformer

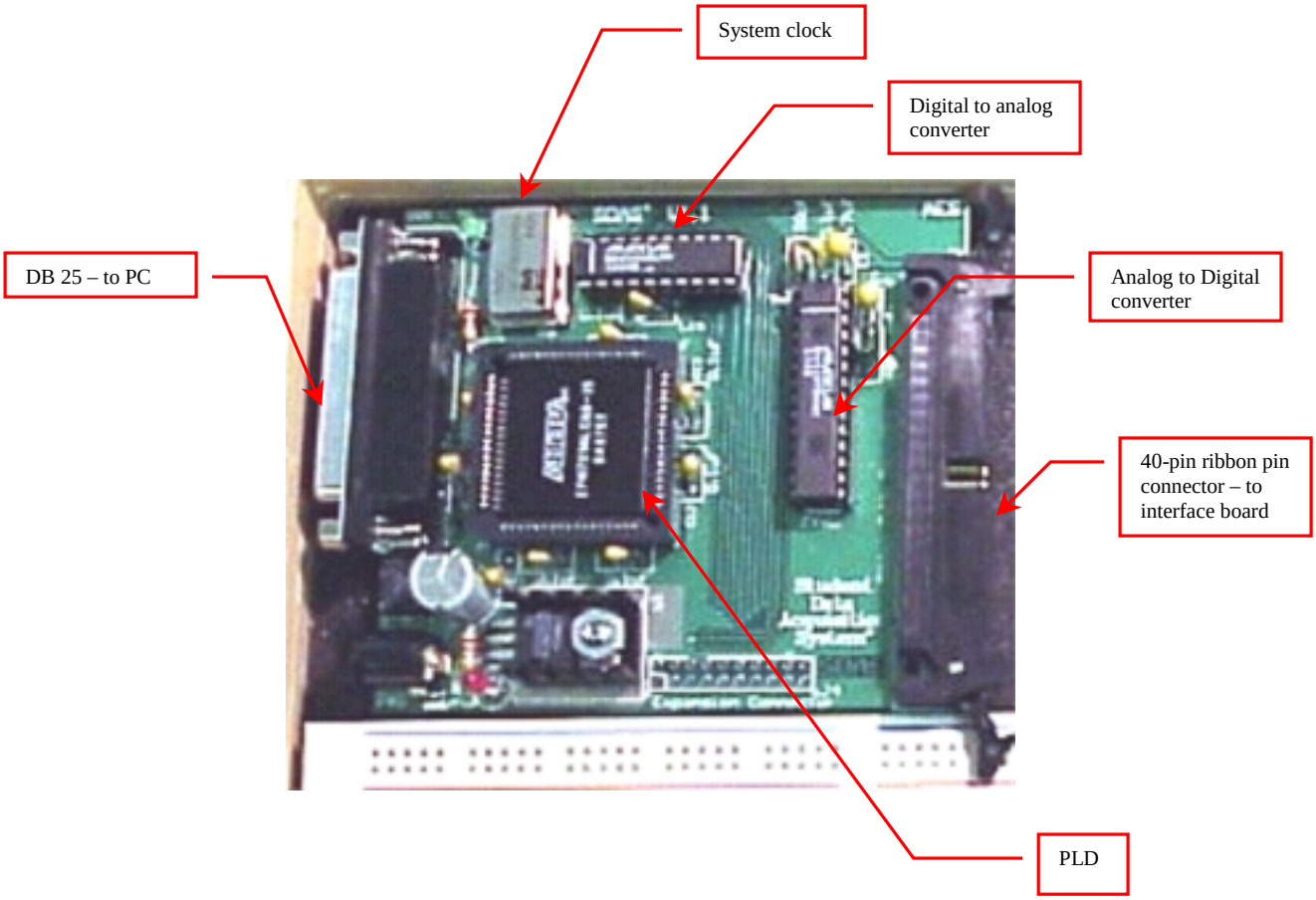

Figure 3 – Major components of SDAS board

Some of the motivation to design a HAC was to exercise my education in heat transfer and in part was inspired by my work at Symyx. I am currently working on a design of a heated wash station for a liquid dispensing robot.
The concept in terms of the heat transfer was quite simple and resulting geometry of the heated block shows this. The heat source is a cylindrical 25 W cartridge heater, .125" in diameter and 1.25" long placed

directly in the center of a 1.25"Aluminum cube. The temperature sensor, an LM35 monolithic IC based temperature sensor was placed in the center of an adjacent perpendicular plane of the cube. The cooling system in the form of the thermoelectric, heatsink and fan was placed over the entire plane of the cube adjacent to the two others (see Figure 5 for clarification). The idea was this:

Predict by using,

$$q'' = -k\frac{dT}{dx}$$, where $k_{Al@300K}$ = 237 W/mK

and symmetry, how quickly the LM35 and thermoelectric will reach the target temperature upon heating. At this point the heat flux of the heater would be turned off and the flux associated with the thermoelectric could be used to predict the cooling rate of the simple block. The rates of heating and cooling could also be modified for other dynamic effects. In addition, the convective cooling of the heat sink could also be analyzed and quantified. Radiation could also be studied by doing the appropriate energy balances. Some of the specifications of the components of the HAC are shown here:

- Heater
  - Watlow, Firerod C1E14, 120 VAC, 25 W, 13 W/cm^2
- Thermoelectric
  - Tellurex CZ1-1.0-127-1.27, $\Delta T_{max@q=0}$ = 79 °C, $q_{max@\Delta T=0}$=38.7 W, q_{typ}=8-17 W, i_{max}=3.9 A, V_{max}=16.1 V
- Fan
  - 5 V, 7 cfm
- Temperature Sensor
  - Calibrated directly in ° Celsius (Centigrade)
  - Linear + 10.0 mV/°C scale factor (Output)
  - 0.5°C accuracy guaranteeable (at +25°C)
  - Rated for full -55° to +150°C range
  - Suitable for remote applications
  - Low cost due to wafer-level trimming
  - Operates from 4 to 30 volts
  - Less than 60 µA current drain
  - Low self-heating, 0.08°C in still air
  - Nonlinearity only ± 1 /4 °C typical
  - Low impedance output, 0.1 W for 1 mA load

The resolution of the sensor with respect to the system was solved in the following way:

$$resolution = \frac{V_{fullrange}}{2^{12}}$$

where $V_{fullrange}$=0.3 V. This gave a resolution of +/- 1.0 mV or +/- 0.1 °C.

The interface circuit to do the switching used simple MOSFETs, DC voltage sources, a relay, a diode, a fuse , the HAC elements and a 120 VAC source. The diagram for this and specifications on the additional components is shown in Appendix B.

The Solution

The hardware solution took the form shown in Figure 4 (minus some additional wiring – Refer again to Appendix B). The physical realization of the HAC took the form shown in Figure5. The first attempt at solving the system and getting it to perform was not trivial. The attempt included using the SDAS board that I had constructed and coding in C to operate the system.

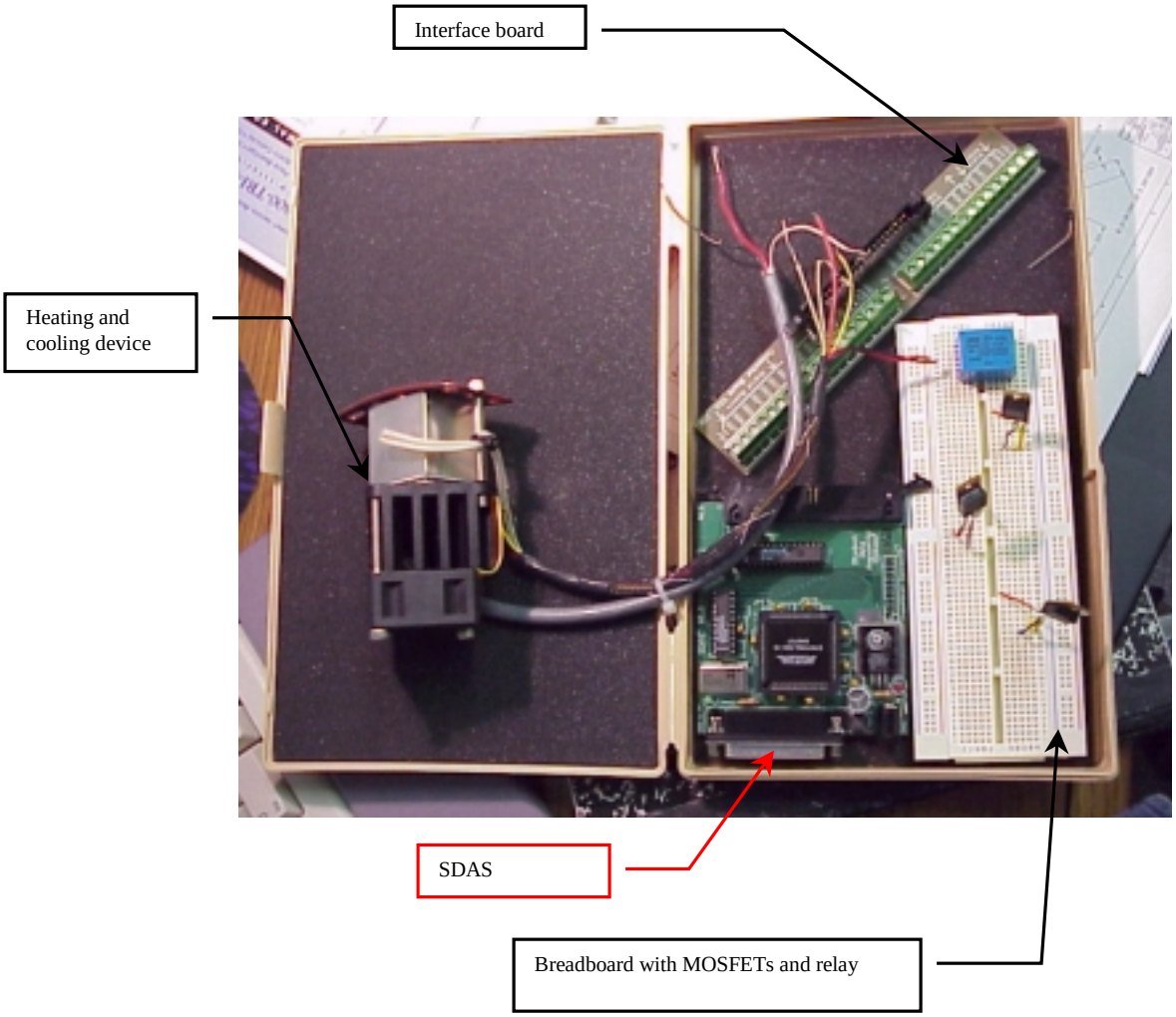

Figure 4 – System components

One of the many attempts to code in C to control the system looked something like the code found in Figure 6.

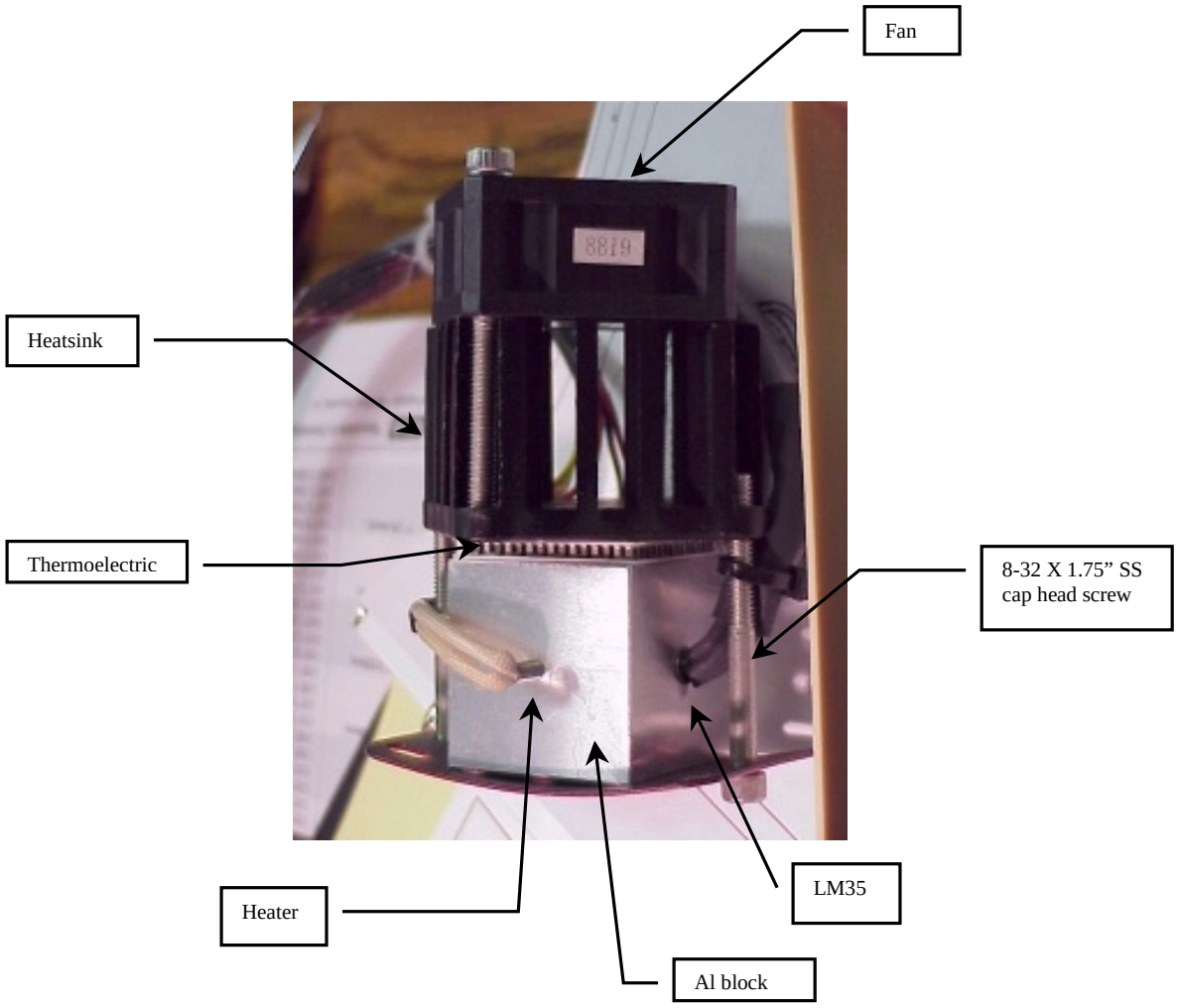

Figure 5 – Elements of the heating and cooling device

In this code, the operation of the system was to heat up to 60 °C then turn the heater off, turn the cooling system on, cool to 30 °C then turn the cooling system off then loop around again. This approach did not work for three reasons: 1) the SDAS board I had built had malfunctioned and 2) the functions located in sdasapi.h would not load on my compiler 3) the A2D conversion may have had to have been in the form of a voltage not a digital value. The compiler used was Borland Turbo C++ 4.5.

Functionality was achieved by coding in QBASIC and switching to a fresh SDAS board. In the source code in Appendix A, the A2D conversion is taken care of in lines 69-98. The digital output on a specific channel is shown in lines 135-139. The main function is found in lines 57-68. The template for this code is found on the SDAS web site (8) and is called sdas_fn.bas. The output of the LM35 was displayed on the screen in a continuous manner as an analog voltage. The voltage displayed came from the LM35’s linear output of 10 mV/°C. For example 60°C converts to 0.60 V.

By using this code, the HAC was controlled by the SDAS through the switching circuitry.

```
# include <stdio.h>
# include <sdasapi.h>

void main (void)
{       int adc, errorcode, errorcode1, errorcode2;
        unsigned int pldv, apiv;
        float lm35;

        errorcode1 = Set_parallel_port();
        errorcode2 = Version_check (&pldv, &apiv);
        if (!errorcode1 && !errorcode2)
        {
                printf("Ready?  Begin by hitting a key.");
                do{}
                while(!kbhit());
                while (1)
                        {       A2D(0-5, 0, lm35);

                                adc=lm35

                                if (adc<492)
                                        {       DigOut (0x1)}
                                else if (adc>246)
                                        {       DigOut (0x6)}

                        }
          }
```

Figure 6 – C code example

Another aspect of the solution came about with the discovery that the computer used to communicate with the SDAS had to have a relatively slow CPU clock speed. The computer used for the solution was a Dell XPS 133 with a clock speed of 133 MHz. Testing runs with computers with higher speeds proved ineffective.

Analysis and Performance Results

Upon achieving the solution, the performance of the system did not hit the desired effect. The QBASIC code did not allow the temperature to fluctuate between 30 and 60 °C. Instead the temperature of the block ramped up to 60 °C and then fluctuated between 59 and a little above 60°C corresponding to output voltages in the range of 0.595 and 0.610 V. The performance of the LM 35 was very good. The resolution of the sensor was evident on the output voltages on screen. You could see the voltage and hear the switching of the relay when the voltage ran around the target temperatures.
The fluctuation at the 60°C had a hysteretic effect on the MOSFET/relay circuit. The relay made these horrible noises as the temperature moved above and below the target parameter.

Another failure was the thermoelectric. The amperage required to run the device was not available with the 12V power supply used. Upon experimentation the amperage output of the power supply was about 0.7 A. The thermoelectric was subsequently cut out of the circuit. However it was left physically in the HAC and seemed to display a relatively good thermal conductivity.

The cooling then was achieved by the use of the fan and heatsink. Because the thermoelectric was apparently conducting heat from the block to the heatsink and the fan was convectively cooling the heatsink, the temperature was indeed able to drop. The response was surprisingly quick.

The proxy SDAS board appeared to perform flawlessly.

The computer in the system worked as long as the CPU speed was adequately slow.

Discussion

The project achieved all of the desired effects it was intended to achieve. Engineering principles such as

- Modes of heat transfer
- A2D and D2A conversion
- Interfacing with a microcontroller or in this case a PLD for a mechatronic application
- Inductance
- Hysteresis effects
- MOS field effect transistors
- Basic AC and DC electronics
- Logical troubleshooting practice
- Temperature effects on NPN transistors
- Written and verbal communications
- Programming
- Patience
- Sensor resolution
- Manufacturing techniques
- Research

were explored and experienced.

Manufacturing turned out to be a key element in the process. Poor practices can lead to disastrous effect. This process should not be take lightly in engineering practice.

The cost breakdown for this project is as follows:

SDAS system	$100.00
Heatsink	$0.95
LM35 temperature sensor	$22.00 (qty 2)
Watlow heater	$24.00 (qty 2)
Fan	$6.00
Relay	$5.28
Wire, fuse holder, extension cord	$12.00
Total:	$216.00

Conclusions and Recommendations

In the future, this project can stand to go through another iteration and refinement of code. As a proof of concept, this system passed. It can interact through digital I/O and perform a task. Because of this fact, the system can refined. Recommendations for this refinement process are as follows:

- Discovery of how to operate the thermoelectric
- Direct attachment of the block to the thermoelectric with some sort of heat conducting epoxy
- Direct attachment of the thermoelectric to the heatsink
- The insulation of the 5 sides of the cube not in contact with the thermoelectric
- Quantification of the heat transfer processes
- Based on these calculations predict the performance of the system
- A continuation of coding in C with the predictions in mind
    - Finding the right compiler
    - Open dialog with Dr. Chris Braun about some of the shortcomings of running the 16-bit C source code functions
- Upon hardening of the system it can be further used for educational purposes
- Turning in this report on a CD because of the graphic content and because I have generated at least 10 pounds of paper for this class – please refer to drawing files on disk for clearer representation of the wiring diagrams

References

1) Histand, M., and Alciatore, D., "Introduction to Mechatronics and Measurement Systems", McGraw-Hill, Boston, 1999

2) Incropera, F., "Fundamentals of Heat and Mass Transfer - 4th edition", John Wiley & Sons, New York, 1996

3) Schneider, D., "Handbook of BASIC – 3rd Edition", Brady, New York, 1988

4) Overland, B., "C in Plain English", MIS:Press, Foster City, California, 1995

5) Hanly, "Problem Solving and Programming Design in C" – 2nd Edition", Addison-Wesley, Reading, Massachusettes, 1996

6) Altera web site: http://www.altera.com/html/products/m7k.html

7) National Semiconductor web site: http://www.national.com

8) Colorado School of Mines web site: http://www.mines.edu/Academic/eng/edu/SDAS

Appendices

Appendix A – Source Code in QBASIC

```
1.  '  Template for use program
2.  '
3.  '  ppId% is the printer data port number   1=&H3BC   2=378   3=278
4.  '
5.  '  The analog to digital converter is set for 0 to 5 volts and A2D% returns
6.  '  a voltage from 0-5 V corresponding to ADC count 0 to 4095.
7.  '  AnaOut is the voltage (0-5V) sent to the selected channel 0-3
8.  '
9.  '  The input operations could possibly "timeout" due to unconnected cables,etc.
10. '  The return value for these functions will be 0-255 for DigIn and
11. '  0-5V for A2D if successful.  If there is a problem, then the
12. '  result is set to -1000
13. '
14. '  C. Braun, Applied Electronics Solutions, 1997
15. '  All rights reserved.
16. ' ============================================================
17.
18. DECLARE SUB DigOut (OutNum%)    ' Outputs a digital value
19. DECLARE SUB D2A (AnaOut, Chan%) ' Outputs analog value to a channel
20. DECLARE SUB ClearOutputs ()     ' Zeros all outputs
21. DECLARE SUB PrintBinary (value%) ' Utility for printing binary pattern of an int
22. DECLARE SUB SetParallelPort ()  ' User dialog to set the parallel port
23.
24. DECLARE FUNCTION DigIn% ()      ' Returns the digital in value (0-255)
25. DECLARE FUNCTION A2D (Chan%)    ' Returns analog volt (0-5V) of a channel
26. DECLARE FUNCTION Version% ()    ' Returns the version # (should be 3)
27.
28. ' Global Values for subroutines
29. COMMON SHARED ppid%     ' parallel port address, set by SetParallelPort or program
30.
31. ' ============================================================
32. ' Loopback program
33. ' Must have a loopback connector installed
34. ' that connects outputs to inputs.
35. ' Program runs through all digital values
36. ' and small steps for the analog values
37. ' to verify that the entire data acqusition system
38. ' is working properly
39. '
40. ' ============================================================
41.
42. ppid% = &H278   ' 1=3bc  2=378  3=278
43.
44. ' ============================================================
45.
46. CLS
47. CALL SetParallelPort ' sets ppid% to user select value
48.
49. ' Do a version check (& resets outputs)
50. ver% = Version%
51. PRINT "The version of the PLD software is "; ver%
52.
53. ' Clear All
54. CALL ClearOutputs
55.
56. ' ============================================================
57.
58. WHILE (1)
59.
```

```
dig# = A2D(0)
IF dig# < .605 THEN DigOut (&H1)
PRINT "Analog output is "; dig#
IF NOT dig# < .580 THEN DigOut (&H6)

WEND
END


FUNCTION A2D (Chan%)
' 0 to 5 volt input range only!
' Returns the voltage, not the count number
' Range is set by configuation value, here 0-5V
OUT ppid% + 2, 11      ' Null Command to SDAS
OUT ppid%, Chan%
' Configuration value 0-5V, channel Chan%  to data register
OUT ppid% + 2, 14
'Now set the configuration Command for the ADC to control register
OUT ppid% + 2, 12       ' Start ADC Command to control register
FOR i = 1 TO 100
temp% = INP(ppid% + 1)           'Read low nib from printer port
IF (temp% AND &H8) THEN EXIT FOR  ' have a handshake for good data
NEXT i
low.nib% = (temp% AND &HF0) / 16  ' Mask junk and shift
OUT ppid% + 2, 3         ' Handshake  for 1st nibble  to control register
FOR i1 = 1 TO 100
temp% = INP(ppid% + 1)              'Read mid nib from printer port
IF NOT (temp% AND &H8) THEN EXIT FOR ' have a handshake for good data
NEXT i1
mid.nib% = (temp% AND &HF0)  'Mask out junk
OUT ppid% + 2, 2        ' Handshake for 2nd nibble to control register
FOR i3 = 1 TO 100
temp% = INP(ppid% + 1)           'Read high nib from printer port
IF (temp% AND &H8) THEN EXIT FOR  ' have a handshake for good data
NEXT i3
high.nib% = (temp% AND &HF0) * 16       ' Mask out junk and shift
OUT ppid% + 2, 11       ' Null Command to control register
A2D = (low.nib% + mid.nib% + high.nib%) / 4095 * 5
END FUNCTION
SUB ClearOutputs
       OUT ppid% + 2, 11       ' Null to control register
       ' Turn off Digital output
       CALL DigOut(0)
       ' Set all Analog Outputs off
       FOR i% = 0 TO 3
       CALL D2A(0, i%)
       NEXT i%
       END SUB
       SUB D2A (AnaOut, Chan%)
       OutValue% = AnaOut / 5 * 255
       OUT ppid% + 2, 11       'Null Command to control register
       OUT ppid%, Chan% ' configuration value for DAC ch 0-3 to data register

       OUT ppid% + 2, 13       ' Configure the DAC Command to control register
       OUT ppid% + 2, 11       'Null Command to control register
       OUT ppid%, OutValue%    ' DAC channel data to data register
       OUT ppid% + 2, 3        ' Start DAC command to control register
       OUT ppid% + 2, 11       'Null Command to control register
       END SUB
       FUNCTION DigIn%
       OUT ppid% + 2, 11       'Null Cmd to control register
       OUT ppid% + 2, 2        'Start Dig Input Command to control register
       FOR i = 1 TO 100
       temp% = INP(ppid% + 1)          'read low nib from printer port
       IF (temp% AND &H8) THEN EXIT FOR  ' have a handshake for good data
       NEXT i
       N0 = (temp% AND &HF0) / 16' shift down
       OUT ppid% + 2, 3        'Handshake - got 1st nibble to control register
       FOR i = 1 TO 100
       temp% = INP(ppid% + 1)  'read low nib from printer port
       IF NOT (temp% AND &H8) THEN EXIT FOR ' have a handshake for good data
```

```
131.       NEXT i
132.       DigIn% = (temp% AND &HF0) + N0
133.       OUT ppid% + 2, 11       'Null Cmd to control register
134.       END FUNCTION
135.       SUB DigOut (NumOut%)
136.       OUT ppid% + 2, 11       'Null Cmd to control register
137.       OUT ppid% + 2, 1        'Start Digital Output Command to control register
138.       OUT ppid%, NumOut%      ' Output data to data register
139.       OUT ppid% + 2, 11       'Null Cmd to control register
140.       END SUB
141.       SUB PrintBinary (value%)
142.       FOR i% = 7 TO 0 STEP -1
143.       bit% = 2 ^ i% AND value%'true if bit set
144.       IF bit% THEN
145.       PRINT "1";
146.       ELSE
147.       PRINT "0";
148.       END IF
149.       IF i% = 4 THEN PRINT " ";
150.       NEXT i%
151.       END SUB
152.       SUB SetParallelPort
153.       quit% = 0
154.       WHILE quit% = 0
155.       CLS
156.       PRINT "Setting the Parallel Port for this program and computer"
157.       PRINT "There may be several possible parallel port addresses:"
158.       PRINT "The current setting is ";
159.       SELECT CASE ppid%
160.       CASE &H378
161.       PRINT "378 hex"
162.       CASE &H278
163.       PRINT "278 hex"
164.       CASE &H3BC
165.       PRINT "3BC hex"
166.       CASE ELSE
167.       PRINT "unknown"
168.       END SELECT
169.       PRINT
170.       PRINT " Your choices are:"
171.       PRINT "    1=3bc  2=378  3=278 ";
172.       INPUT choice%
173.       SELECT CASE choice%
174.       CASE 1
175.       ppid% = &H3BC
176.       CASE 2
177.       ppid% = &H378
178.       CASE 3
179.       ppid% = &H278
180.
181.       CASE ELSE
182.       BEEP
183.       END SELECT
184.       PRINT
185.       PRINT "Now check for communications to the SDAS (make sure system on)"
186.       PRINT "Press space-bar to toggle the Green LED"
187.       PRINT " or Y for Green LED flashing, N if not flashing or Q to quit"
188.       quit1% = 0
189.       WHILE quit1% = 0
190.       keyin$ = UCASE$(INPUT$(1))
191.       SELECT CASE keyin$
192.       CASE "Y"
193.       PRINT "Good!  You have set the parallel port correctly"
194.       quit% = 1
195.       quit1% = 1
196.       CASE "N"
197.       PRINT "Try another parallel port address please"
198.       quit1% = 1
199.       CASE "Q"
200.       quit1% = 1
201.       quit% = 1
```

```
202.        CASE ELSE
203.        IF outbyte% = &H80 THEN outbyte% = &H0 ELSE outbyte% = &H80
204.        OUT ppid%, outbyte%
205.        END SELECT
206.        WEND ' quit1%
207.        OUT ppid%, &H0 ' turn off
208.        WEND ' qui
209.        ver% = Version% ' update version
210.        END SUB
211.        FUNCTION Version%
212.        OUT ppid% + 2, 4        'Reset/version Command to control register
213.        temp% = INP(ppid% + 1)
214.        temp% = (temp% AND &HF0) / 16 ' Mask out junk
215.        OUT ppid% + 2, 11       'Null Command to control register
216.        Version% = temp%
217.        END FUNCTION
```

Appendix B – System Wiring Diagram