EECE 281 - ELECTRICAL AND COMPUTER ENGINEERING DESIGN STUDIO

PROJECT 1 - REFLOW OVEN CONTROLLER

INSTRUCTOR: JESUS CALVINO-FRAGA

SECTION: LAB L2A

GROUP NUMBER: AC

DATE: FEB 24, 2014

GROUP MEMBERS

Wyatt Gronnemose	33865122
Connor Rambo	19352137
Balpreet Parhar	43314129
Samson Gama	30550123
Albert Hynek	34224121
Shivanshu Goyal	33245127

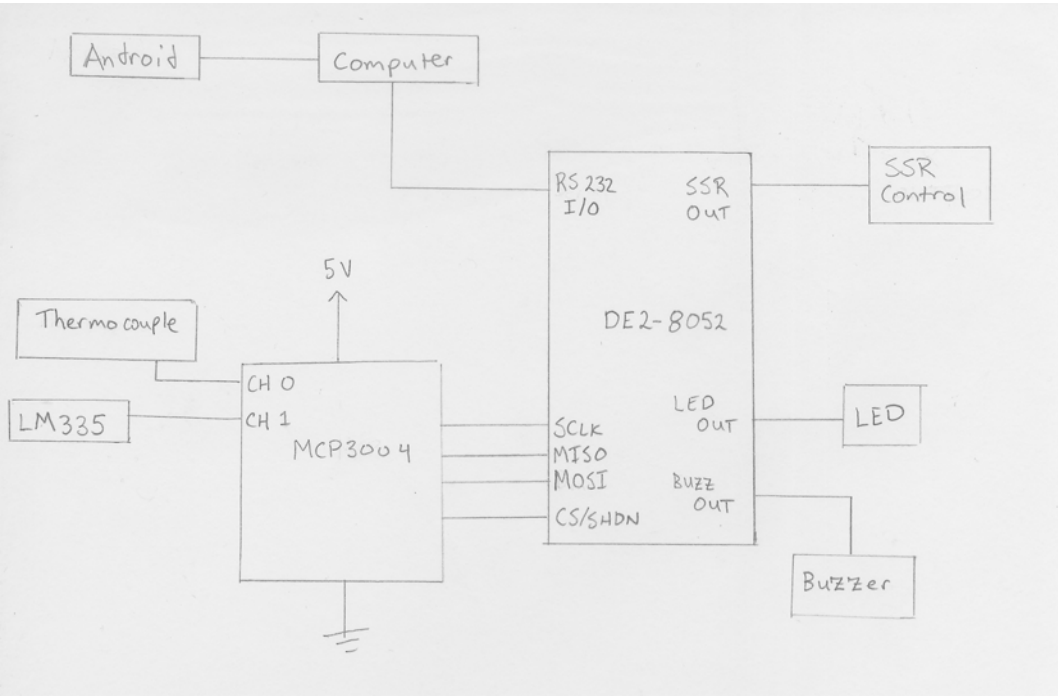
TABLE OF CONTENTS

1.0 INTRODUCTION

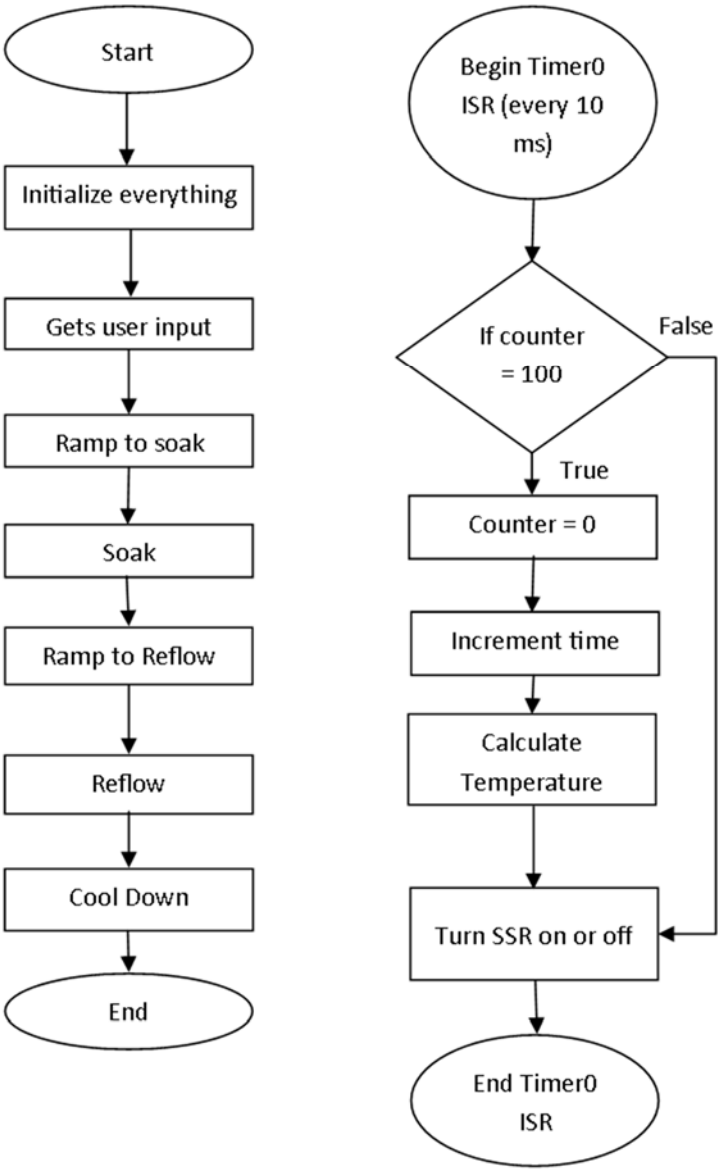
The goal of this project is to construct a reflow soldering oven controller using an Altera DE2 FPGA loaded with a DE2-8052 soft-processor. Reflow soldering is a technique used to solder surface mount devices (SMD's) onto printed circuit boards (PCB's). Prior to the reflow process itself, a PCB must be prepared by to covering the pads on the PCB with solder paste (a mixture of flux and powdered solder) and carefully placing the components on top. The PCB is then heated in an oven, where the reflow process is controlled by the oven controller. When the solder paste becomes hot enough, the solder powder will melt and attach the components to the PCB. During the reflow process, the oven controller can be in one of five states: ramping to soak (constant rise to a given temperature), soak (maintaining given temperature for specified period of time), ramping to reflow (see ramping to soak), reflow (see soak) and cooldown (oven is off). The soak time, soak temperature, reflow time and reflow temperature are all configurable by the user. Throughout the process, the current temperature is to be displayed on a strip chart and updated every second.

We approached the problem by first brainstorming ideas and drawing out rough top level block diagrams and flowcharts, and then slowly designing and refining the intricacies of each component in the block diagram. All special features that were implemented were part of the design process from the very beginning, and their implementation, as well as the implementation of base project requirements, were kept in mind throughout the design and implementation phases.

HARDWARE BLOCK DIAGRAM



SOFTWARE BLOCK DIAGRAM



2.0 INVESTIGATION

2.1 IDEA GENERATION

Our team generated ideas by meeting regularly, discussing project requirements and creating rough design documents. We divided into hardware and software teams, but continued collaborating with each other. Each group had to create a flowchart, which allowed for further analysis of the problem.

For hardware, the necessary parts for the circuit had to be determined. These included the Altera DE2 Board, LM335 temperature sensor, MCP3004 analog-to-digital converter (ADC), OP07 operational amplifier, LMC7660 voltage converter, PN2222 transistor, solid state relay (SSR), piezoelectric buzzer, k-type thermocouple wire and light emitting diode (LED) .

The software was programmed based on the foundations of inputs and outputs. This allowed an understanding of the information from the hardware and we determined what needed to be produced. We concluded that the inputs were amplified thermocouple voltage and LM335 temperature reading; the outputs were SSR control, buzzer control, LED control, and serial information to the computer. As a group we discussed required data manipulation to achieve the correct result, such as lookup table conversions and mathematical ADC temperature conversions.

2.2 INVESTIGATION DESIGN

Our group's design investigation consisted of information and data gathering, analysis, and experimentation. The hardware component of this project dealt with the data acquisition, while the software component dealt with analysis.

For the hardware design portion, we started by drawing a grid with the same dimensions as our protoboard. We strategically placed major components, to minimize crossed wires. To ensure no mistakes, we used various datasheets to confirm pin alignments and port specifications.

The two hardware components that acquired the data for our project were the thermocouple wire and the LM335. The values from these components were manipulated and added together in software.

All analysis occurred in the software., we needed to design the state transitions. Our program analyzed the data and responded depending on the current state, user inputs, and data readings. We had to design the states in such a way so that they would transition as specified times and

temperatures were reached.

We also did some experimentation. To fully understand the accuracy and utility of components in our project, testing was required. For all of us, thermocouple technology was a new concept. We tested the thermocouple using a soldering iron in the lab to see if the digital value from the ADC would increase as it heated up. We then incorporated the code to convert the digital value to a temperature, and used a toaster oven and a thermometer to make sure that we were measuring temperature within a couple of degrees. We also used the appropriate lab equipment to test the LM335, MCP3004, LMC7660, and OP07, to ensure the chips were fully functioning.

2.3 DATA COLLECTION

In order to make an accurate fully functioning Reflow Soldering Oven Controller, we had to manipulate our inputs to appropriate real world outputs. There were two primary external inputs that we needed to collect and analyze data from: the thermocouple and the LM335 temperature sensor. The thermocouple voltage needed to be amplified to give the data a greater range. From here, the analog signal of the thermocouple needed to be converted through the MCP3004 ADC; the LM335's readings were also fed into the ADC.

Thermocouple technology was a new concept for our group. We learned that changes in temperature resulted in changes in potential difference at the end of the thermocouple. To ensure the technology was accurate and not faulty, we did a series of tests measuring the potential difference across the wires at different temperatures using soldering irons and heaters. We used a datasheet, as well as comparison readings with the LM335, to verify the readings were accurate. During our tests, we learned that if the thermocouple came in contact with another metal, such as the oven rack, it could cause an unexpected voltage spike. Keeping this in mind, we made sure to take proper care to avoid such occurrences when operating our reflow oven.

When designing the circuitry that would measure the temperature and control the oven, we first breadboarded the circuit. While it was on the breadboard, we checked the voltage outputs of the op amp, LM335 and negative voltage generator. Since the actual project was not complete at that point, we also wrote some basic test code that would demonstrate we were acquiring the digital output of the ADC correctly, and that it corresponded to the correct temperature. This ensured the accuracy of the MCP3004. Finally, we did complete tests by outputting the summation of the cold and hot junctions on the seven-segment displays. Again, we tested this by applying varying

temperatures, such as using a soldering iron to see increases in the temperature readings. By verifying accuracy of the technology, we felt confident in finalizing and soldering our designs onto a protoboard.

2.4 DATA SYNTHESIS

In order to output the correct results, we needed to analyze and modify the input data. Both the thermocouple and LM335 Temperature Sensor readings needed to be mathematically converted to appropriate real-world values. The thermocouple is first amplified by a scaling factor of 488, then alongside the LM335's reading, it is fed into the ADC. The values are altered using relevant formulae in software. From here, the values are comprehended by a lookup table. Finally, the data is combined by software addition. At this stage, the synthesized value is used for state transition boolean expressions.

2.5 ANALYSIS OF RESULTS

To confirm the accuracy of our results, we displayed the LM335 and thermocouple readings on the hex displays. In a typical test, the LM335 displayed a reasonable room temperature (between 22^{o}C and 25^{o}C) and the thermocouple displayed zero because both ends of the wire were in the same environment.

Attaching the thermocouple wire to the breadboard introduces extra thermocouple contacts. Due to this, an extra junction is required in the wire. Temperature is measured at each junction and thus, requires two temperature measuring devices. The use of multiple components inherently increases the margin for error, as the individual error of each component contributes to the overall error. The temperature measurement could have been more accurate, for example, if measured solely with the LM335.

Further discrepancy in temperature could be due to an approximation made on the thermocouple voltage. Real-world thermocouple voltage does not change linearly with temperature, however, our lookup table was calculated linearly[2]. This approximation, however, is very close to real world behaviour and has a very minor effect on temperature reading .

Lastly, is the operational amplifier. The thermocouple wire produces a very small voltage change per degree (about 41μV/^{o}C). This meant that we had to use an op amp with a very low input offset voltage, such as the OP07, to amplify our input[3]. If a more basic op amp had been used, it

would not have been able to tell that the small differences in the input actually constituted a temperature change. However, even with the OP07, some amount of error will result when trying to amplify a tiny signal to something that is easier to measure.

3.0 DESIGN

3.1 USE OF PROCESS

Our group made good use of key design principles to maximize the efficiency of our work, to reduce the pain of integration, and to ensure maximum robustness and functionality.

In the initial design phase, our team collaborated on the creation of various design documents, including flowcharts, software and hardware block diagrams, state transition diagrams, and basic mathematical algorithms. These documents helped ensure all members of the group were aware of the overall scope of the project, as well as how each individual component and feature of the project would be implemented. Major special features were decided upon and kept in mind throughout the project's development, right from very beginning.

In order to utilize the diversity of our group members' respective expertise, individual members were assigned portions of the project that best suited their interests and technical abilities. For example, those in Electrical Engineering focused mostly on the circuit design, while Computer Engineering students focused on the software design and primary special feature. That being said, all members contributed significantly to the main assembly codebase due to everyone's familiarity programming the 8052 microcontroller.

Throughout the project, all work was stored on and contributed to using collaborative tools such as GitHub and Google Docs. Any changes made by any member of the group were immediately visible and accessible to all members, vastly improving our collective efficiency and minimizing the modifications required when all modules were brought together.

3.2 NEED AND CONSTRAINT IDENTIFICATION & PROGRAM SPECIFICATION

Beyond the original project requirements[1], our group took into account a variety of stakeholder needs in all of our design decisions. During the design phase, we made the determination that we should strive for the easiest possible way for users to input parameters and operate the reflow oven. As a result, we managed to keep manual inputting of parameters to only two pushbuttons. Additionally, we decided to make operation of the reflow oven possible remotely and visually through an Android application. Keeping in mind the need to not use any extra hardware, we made sure to implement all aspects of this communication exclusively in software.

In order to support two-way serial communication, as well as other design considerations such as platform portability and ease of use, we decided to develop from scratch a desktop application in Java to handle our additional functionality in addition to the required stripchart capture. Anticipating enterprise needs, our Java application allows us to export all captured stripchart data into a .csv file that can later be graphed and manipulated using software such as Microsoft Excel.

Finally, certain potential aspects of the project that were deemed to not be required, such as the handling of negative temperatures, were cut from the project, allowing us to save time and focus our resources on more relevant and immediately useful pieces of functionality.

A more formal representation of certain requirements based on decided-upon special features are shown as follows:

Additional User Interface & Feedback Requirements:
The Reflow Oven Controller must have the added functionality:

a. Selectable reflow profile parameters such as soak temperature, soak time, reflow temperature, and reflow time. These or equivalent parameters should be selectable using the switches and pushbuttons available in the DE2 board and the Android application
b. Collected temperature readings must be graphed in real-time on the server, made available to the android app, and should be easily exportable for graphing software.

3.3 SOLUTION GENERATION

Designs for getting inputs:

- Using a push button to increment values, and another push button to decrement values shown on the LCD screen. This was chosen to make it easy for the user to choose precise values for setting temperatures and times. Another push button was chosen to choose different setting options, example going from setting soak time to setting soak temperature, etc.
- The Android app's primary focus was to also to make inputting parameters as simple as possible for the user, as well as the ability to monitor current data
- We started working on an extra feature for getting the inputs on the DE2 board. Pressing the push buttons for incrementing/decrementing values would slowly increase the speed of changing values, so the user would not have to wait long to get to large values starting

from small values, and vice versa. However, we did not implement this feature because we were running out of time, and we already had the Android app which could be used for inputs.

Designs for outputs:

- We chose to show the current temperature and elapsed time on the LCD at all times once the reflow process has been started. The desired temperatures and times are shown on the seven segment displays.
- The current temperature and current state are also shown on the Java app (running on the computer) both as a real-time status line, and in the form of a dynamically-resizeable strip chart
- The current state, temperature and elapsed time are shown on the Android App as well
- All captured strip chart data (temperature and time) can be exported to a file for further analysis

Design for controlling the oven:

- The controller for the oven was designed as a state machine which transitions from one state to the other. It has six states - waiting for inputs, ramp to soak, soak, ramp to reflow, reflow, cool down.
- When in a steady state (either Soak or Reflow), the oven is kept at a steady temperature by cycling the oven using pulse wave modulation for 10 percent power
- To help prevent unwanted temperature overshoot in the oven, we designed a system that turns off the oven during a ramping state well before the desired temperature has been reached without causing a state transition

Design for communication between the Java application and the DE2-8052:

- All the communication between the Java application and the DE2 happened using serial (RS-232 standard). Since hardware handshaking is not implemented on the DE2-8052, basic software handshaking had to be designed for bidirectional communication.
- We used an open-source Java serial communication library[4] for all serial communication. This is a platform-independent library which suited our needs of making the Java application platform independent.

Design for communication between the Android app and the Java app:

- The communication between the Android app and the Java app was accomplished

through the use of an embedded web[5] and PHP[6] server, and an SQLite database, all of which were embedded in the Java application. We wrote a PHP script which wrote reflow parameters into the SQLite database. When the submit button in the Android application is pressed, this PHP script is called with the reflow parameters by means of an HTTP request. When the reflow process starts, the Java application reads from the database, sending all reflow parameters to the DE2. The application also writes data such as the current oven temperature, elapsed time and the current state of the reflow process into the SQLite database into a different table, which is subsequently read by a second PHP script called by the Android application displaying this data in real time.

3.4 SOLUTION EVALUATION

Nearly all of the designs mentioned in the 'Solution Generation' section above were implemented in the final project. Having the bonus features such as the Android app and the Java app in mind from the beginning helped us get a high level overview of the whole system.

- **Getting the inputs:** Having a simple and precise way of obtaining inputs from the beginning proved invaluable when performing tests. Not having to constantly fiddle with imprecise input methods made repetitive testing significantly less difficult, and we can conclude that having multiple precise input methods was the right design choice and helped us exceed project criteria.
- **Displaying the outputs:** We had multiple innovative ways of showing the outputs. The outputs were shown on the DE2 board using the LCD screen and the seven segment displays. The outputs on the Java app were shown through the strip chart, whose domain and range could be modified in real time. The current temperature, elapsed time, and the current state of the reflow process were also shown by both the Java and Android applications. The ability to export the temperature data from the Java app in a CSV file format was also very helpful, enabling us to analyze this data in the future. Overall, this part of the project exceeded the project requirements as well.
- **Controlling the Oven:** As discussed in the previous section, the controller was implemented using a state machine. A lot of thought was put into determining when the transitions between states should take place, in order to get the best possible temperature graph. For example, we made the controller turn the oven off when the temperature reached a few degrees below the soak temperature, to prevent the overshoot of temperature. A kill switch was also implemented for safety purposes.

- **Communication between the Java application and the DE2-8052:** One bonus features we implemented in this part of the project was that the Java application gives the user a list of available COM port in Windows and Serial Ports in Linux / Mac OS. The serial communication was two way. The Java application sent reflow parameters to the DE2-8052 and received other data such as current temperature, time, and state.
- **Communication between the Android application and the Java application:** Embedding all of the server software in the Java application made this communication link very robust and secure.

3.5 DETAILED DESIGN

Our group analyzed a variety of different solutions for each "block" in the [Hardware Block Diagram on pg. 3]. We made prototypes and tested out many solutions before implementing the final design for every block. The blocks from the Hardware Block Diagram are described in detail below:

8052 Microcontroller: The microcontroller was responsible for the following purposes:

- Controlling the oven (using Pulse-Width Modulation) based on the initial parameters (soak temperature, soak time, reflow temperature, and reflow time):
  During the ramp to soak and ramp to reflow states, the oven is simply turned on and allowed to reach soak temperature minus 25 degrees Celsius (to reduce oven temperature overshoot). During the soak and reflow states, the temperature is maintained by use of PWM, turning the oven on for one-tenth of a second every second. For the cooldown state, the oven is simply turned off.

- Getting the inputs for the initial parameters:
  When the DE2 first boots up, the user is given an option to either enter the settings through the Android application or by using the push buttons on the board. If the Android option is chosen, the DE2 stores the parameters as they appear in the SQLite database and skips the manual input steps

- Controlling other peripheral devices such as the LCD and 7-segment displays which displayed the initial parameters and current data (temperature, elapsed time and current state). It also controlled the buzzer and the SSR control LED:
  The buzzer went off between every state change. There was a long beep to indicate when

the user should open the door and 6 short beeps when it is safe to touch.

The SSR control LED is synchronized with the oven control bit

- Communicating with the computer (using RS-232) in order to receive the initial parameters from the computer and send current data to the computer

Computer: The computer ran a Java program responsible for the following purposes:

- Communicating with the 8052 microcontroller through the RS-232 serial communication standard
- Communicating with the Android app using an embedded web[5] and PHP[6] server, which are integrated into the Java application
- Displaying the current information (temperature, time, current state)
- Displaying a temperature/time strip chart
- Exporting the temperature data in a CSV file format

Android: The android app has the following functionality:

- Communicate with the Java application using HTTP requests to the server to send inputted reflow parameters and to receive current data (temperature, time and state).
- Set the initial parameters to initialize the reflow process
- Show and update the current information periodically

3.6 SOLUTION ASSESSMENT

The Reflow Oven Controller had the following requirements:

- Measure the internal temperature of the toaster oven
- Receive user inputs of time and temperature for both the soak and reflow stages
- Transition through a full reflow cycle: ramp to soak, soak, ramp to reflow, reflow, cooldown
- Buzzer must produce a short beep once between every state transition, long beep when it enters the cooldown stage, and six short beeps when the PCB is safe to touch
- Display the state status, time, and temperature on the LCD screen
- Display the SSR status through a LED
- Have a kill switch that is capable of turning off the oven at any moment
- All code for the microcontroller had to be completed in assembly

To ensure accurate results, we implemented many test phases throughout our project process. We

tested many of the various components individually.

For the temperature sensor and thermocouple, we used similar tests. We displayed the LM335 and thermocouple readings on the hex displays. In a typical test, the LM335 displayed a reasonable room temperature (between 22ºC and 25ºC) and the thermocouple displayed zero because both ends of the wire were in the same environment.

For the operational amplifier and voltage converter, we applied test voltages and inspected the output. We did the math calculations to verify correct output values for the operational amplifier. The voltage converter was much easier - we had to check for opposite polarity of the output.

Once we tested the components individually, we inspected the design as a whole. We wrote basic testing code that would display the temperature to make sure that the system was still measuring the temperature correctly. After the software was done, we did tests to see if the states transitioned properly. Once that was working, we fine tuned the PWM to make sure that we did not overshoot the target temperature by too large of a value.

We assessed our oven performance by comparing our reflow plot to an ideal reflow plot. We managed to refine our process until the plots were nearly identical. Our test PCB did not suffer any damage and was soldered as intended, proving itself to work correctly by flawlessly running a blinking light test program.

Regarding information presented to the user, we ensured that the same temperature, time, and state values were displayed on the DE2, the Java application, and the android device and that the buzzer reacted at the correct state transitions and for the correct duration.

Overall, the Reflow Oven Controller had acceptable error and met or exceeded all project goals, presenting a number of strengths and only minor weaknesses.

A key strength of our project was its ability to be controlled from multiple platforms including the Altera DE2 board itself and various Android devices. Further platforms could be supported with ease, allowing for integration into any industry environment, including one largely reliant on automation. The Java application, also multi-platform, also has the benefit of being able to store all recorded data in a CSV file (see Appendix, page 75) for further analysis, manipulation, or storage.

One of the weaknesses of the controller was that whenever the exposed end of the thermocouple

touched metal it would record a momentary downward spike in temperature. Further, we found that different ovens heat up at (sometimes widely) different rates, sometimes causing issues with oven temperature overshooting the target temperature.

Finally, due to a lack of hardware handshaking implemented into the DE2 board's serial port, we had to resort to primitive software-based handshaking, and, as a result, the oven controller must remain in constant serial communication to function properly during the reflow process. For this reason, serial communication should be disabled by use of a switch if the controller is required to function independently.

4.0 LIVE-LONG LEARNING

The courses which we took at our time in UBC Engineering gave us a solid foundation to successfully complete this project. Given that most of the members had extensive experience with programming and electronics outside of academics, we were able to meet and exceed the expectations and goals of the project. However, every group member did not have the same experiences in the past which meant that we all had something to teach another. Apart from the Altera DE2 board and assembly which skills we all possessed, the prototyping, Java, JavaFX, PHP, and Android development skills were skills that at most two of the members knew at the beginning of the project.

Any knowledge gaps, such as integrating Java and Android applications with an SQL database and communicating bidirectionally in serial through Java, were filled in using print and online resources.

In terms of prototyping, the challenge of amplifying the voltages of the thermocouple with the use of op-amps proved to be an exciting problem. Without prior labs in EECE 281 and course material in EECE 251, the prototyping of the circuit would have been nearly impossible.

Without these experiences and available resources, the marriage between the hardware and software would have taken considerably longer, and may easily have required significant expertise not otherwise found within our group.

5.0 CONCLUSIONS

For our Reflow Soldering Oven Controller, we have managed to satisfy and surpass all the basic requirements of a functional reflow oven while also implementing a framework that allows for expandability in the future. The user can set the soak temperature, soak time, reflow temperature, and reflow time using the push buttons on the Altera DE2 board as required or, alternatively, use our OvenController Android application to set and monitor these parameters. At any point during the reflow process, the user can view the current time and temperature as well as the current reflow stage on both the DE2 board and in the Android application. If an emergency stop is needed, the user may kill the reflow process at any time by flipping SW0 on the DE2 board. Using the data we made available from the controller and the framework we built, it would also be very simple to add a kill switch or any other desired function into both the Android application and the Java application in the future.

On the PC monitoring side, we are able to capture the temperature of the oven as it is sent serially from the DE2 board and graph it using JavaFX. Any data that is captured can immediately be exported directly from our Java application into a .csv file, which can later be imported using software such as Microsoft Excel for further analysis and graphing. During capture, all data is also stored in an SQL database, and an embedded server allows our Android application to receive and transmit data to/from the oven controller itself. This may be the most useful feature of our project, as this functionality can easily be extended beyond a local network, allowing our controller to be monitored and operated from anywhere in the world. Such a feature could prove to be invaluable in industry, where prototypes could be built and soldered in mass quantities completely off-site. This also opens the door to integrating our oven controller into much larger automated systems. Importantly, all of this functionality is built into a single, self-contained Java application (complete with serial port selection) that is compatible with both Windows and Mac OS X and is contained in a complete GUI.

By far the most time-consuming and head-scratching problem we encountered was unknowingly fully occupying the lower 128 bytes of RAM. This caused our program to occasionally display garbage values, return to seemingly random places in our code, and would never allow us to proceed past our input state as our program restarted and reset the stack pointer each time. We were stumped on this problem for nearly a week, chasing seemingly unrelated problems to the

point where we seriously considered scratching all our code and starting anew. Thankfully, one of our group members determined we had run out of directly-addressable RAM, and after a few modifications the project went along smoothly. Although we had wasted nearly a week, the good programming practices we followed (namely encapsulation and testing modules in isolation) allowed us to debug quickly and efficiently, and the project was back on track in no time.

On the hardware side, we found that the thermocouple did not work properly whenever the exposed end touched metal. We believe that it was caused by the two wires in the thermocouple shorting, causing there to be no voltage drop across the two ends. We found this problem to be avoidable with proper care, however any instances of thermocouple touching metal would result in unwanted (and often infinitesimally short) downward spikes in reported temperature. Unfortunately, this issue manifested itself during our demonstration, where a slight movement of the thermocouple on the PCB caused the reported temperature to momentarily drop to within safety limits prematurely. This would not be an issue in an industrial setting, as there would need to be an insulating cover over the end of the thermocouple to prevent shorting or damage.

In total, we estimate the project to have taken approximately 30 hours from each member as while our ambitions were great, our issue with insufficient memory really took a toll on both our time and spirits. Once that obstacle was overcome, the project came together wonderfully for a near perfect demonstration, and every member of our team has gone away proud of what we managed to accomplish.

REFERENCES

[1] Calvino-Fraga, Jesus, “Project 1 - Reflow Oven Controller”, EECE 281, January 2014.

[2] “NIST ITS-90 Thermocouple Database”, http://srdata.nist.gov/its90/download/type_k.tab

[3] “Analog Devices Ultralow Offset Voltage Operational Amplifiers”, http://web.mit.edu/6.301/www/OP07_a.pdf

[4] “serial-comm, Platform-independent serial port access for Java”, https://code.google.com/p/serial-comm/

[5] “Jetty Web server and javax.servlet container”, http://www.eclipse.org/jetty/

[6] “Java Implementation of PHP 5”, http://quercus.caucho.com/

BIBLIOGRAPHY

Huang, Han-Way. *Using the MCS-51 Microcontroller.* Oxford University Press, 1999. Print.

Juneau, Josh, et al. *Java 7 Recipes: A Problem-Solution Approach.* Apress, 2012. Print.

Phillips, Bill and Hardy, Brian. *Android Programming: The Big Nerd Ranch Guide.* Pearson Technology Group, 2013. Print.

Tatroe, Kevin MacIntyre, Peter and Lerdorf, Rasmus. *Programming PHP.* O'Reilly Media, 2013.

Print.

"Thermocouples." *Stern Technologies.* Stern Technologies, 10 Jun, 2013. Web. 30 Jan, 2014. <http://www.sterntech.com/pdfs/thermocouples.pdf>.

Weaver, James, et al. *Pro JavaFX 2: A Definitive Guide to Rich Clients with Java Technology*. Apress, 2012. Print.

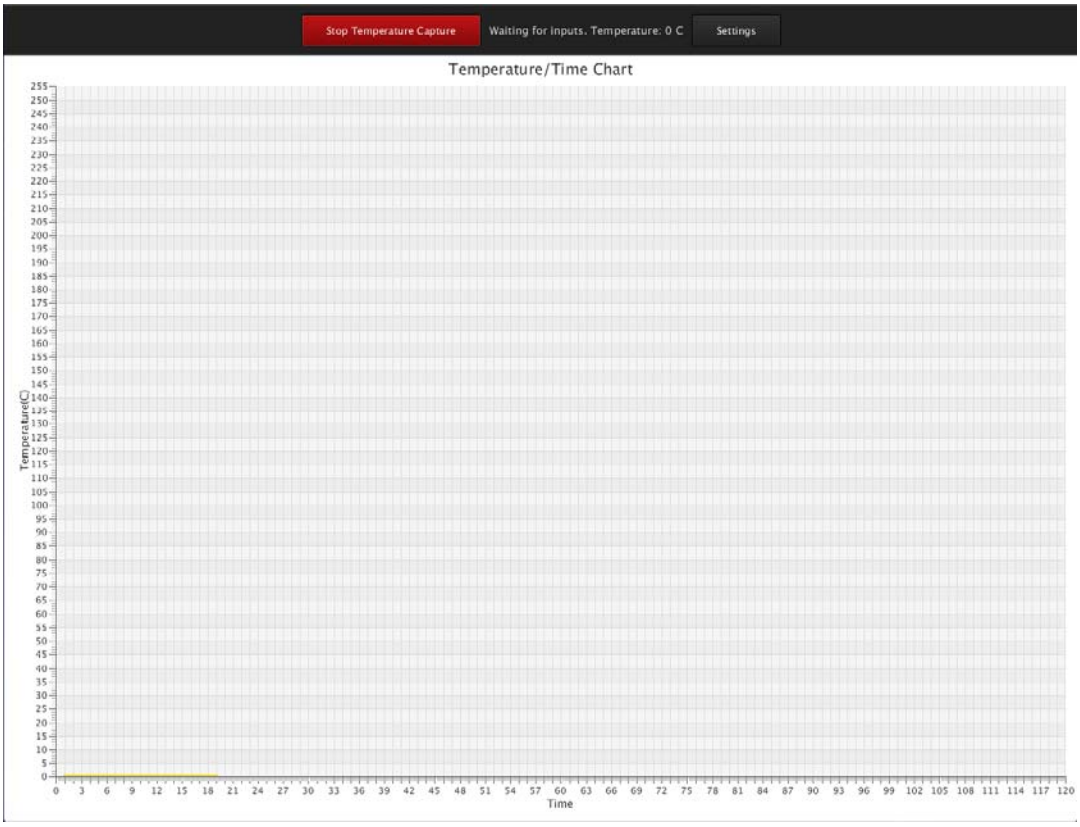
APPENDICES

Exported CSV File Sample (Full Reflow Process)

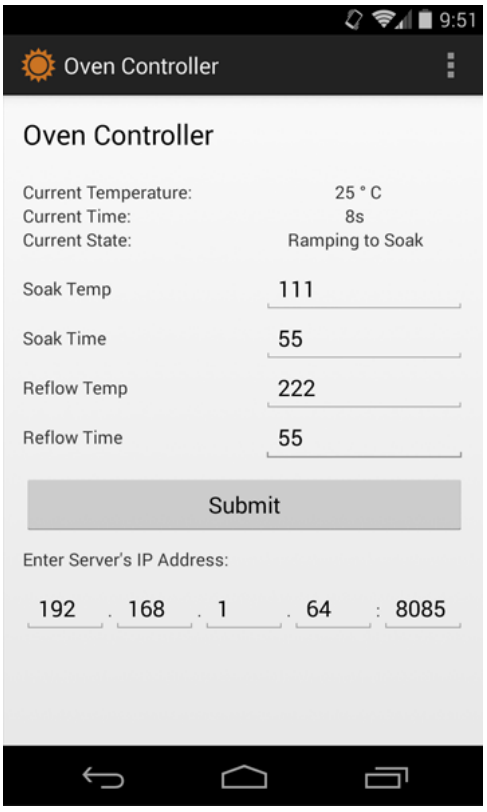
Time, Temperature	60,124	121,148	182,139	243,140
0,68	61,125	122,147	183,139	244,141
1,67	62,127	123,147	184,139	245,142
2,68	63,128	124,147	185,139	246,143
3,67	64,129	125,147	186,139	247,144
4,67	65,131	126,147	187,138	248,144
5,67	66,132	127,147	188,138	249,145
6,67	67,133	128,146	189,138	250,146
7,67	68,134	129,146	190,138	251,147
8,67	69,135	130,146	191,138	252,148
9,67	70,136	131,146	192,138	253,149
10,68	71,137	132,146	193,138	254,150
11,68	72,138	133,146	194,138	255,151
12,68	73,139	134,146	195,138	256,152
13,69	74,140	135,145	196,138	257,153
14,69	75,140	136,145	197,137	258,154
15,69	76,141	137,145	198,137	259,155
16,69	77,142	138,145	199,137	260,157
17,70	78,142	139,145	200,137	261,158
18,70	79,143	140,145	201,137	262,159
19,71	80,144	141,144	202,137	263,160
20,72	81,144	142,144	203,137	264,162
21,72	82,145	143,144	204,137	265,163
22,73	83,145	144,144	205,137	266,164
23,74	84,146	145,144	206,136	267,165
24,75	85,146	146,143	207,137	268,167
25,75	86,147	147,143	208,136	269,168
26,75	87,147	148,143	209,137	270,170
27,77	88,148	149,143	210,136	271,171
28,78	89,148	150,143	211,136	272,172
29,78	90,148	151,142	212,136	273,174
30,80	91,148	152,142	213,136	274,175
31,81	92,149	153,142	214,136	275,176
32,82	93,149	154,142	215,136	276,178
33,83	94,149	155,142	216,136	277,180
34,84	95,149	156,142	217,136	278,181
35,85	96,149	157,142	218,136	279,182
36,87	97,149	158,142	219,135	280,184
37,88	98,149	159,142	220,135	281,186
38,89	99,149	160,141	221,135	282,187
39,91	100,149	161,141	222,135	283,189
40,92	101,150	162,141	223,135	284,190
41,93	102,150	163,141	224,135	285,192
42,95	103,149	164,141	225,135	286,193
43,96	104,150	165,141	226,135	287,195
44,98	105,150	166,140	227,135	288,196
45,99	106,149	167,141	228,135	289,198
46,101	107,149	168,140	229,135	290,199
47,102	108,149	169,140	230,135	291,201
48,104	109,149	170,140	231,135	292,202
49,105	110,149	171,140	232,136	293,204
50,107	111,149	172,140	233,136	294,205
51,108	112,149	173,140	234,136	295,207
52,110	113,149	174,140	235,136	296,208
53,112	114,149	175,140	236,137	297,210
54,114	115,148	176,139	237,137	298,211
55,115	116,148	177,140	238,137	299,213
56,117	117,148	178,139	239,138	300,214
57,119	118,148	179,139	240,138	301,216
58,121	119,148	180,139	241,139	302,217
59,122	120,148	181,139	242,140	303,219

304,220
305,222
306,223
307,224
308,225
309,226
310,227
311,228
312,228
313,229
314,230
315,230
316,231
317,231
318,232
319,232
320,233
321,233
322,233
323,234
324,234
325,234
326,234
327,234
328,234
329,234
330,234
331,234
332,234
333,234
334,234
335,234
336,234
337,234
338,233
339,233
340,233
341,233
342,233
343,232
344,232
345,232
346,233
347,237
348,231
349,224
350,217
351,210
352,205
353,201
354,196
355,192
356,187
357,183
358,180
359,176
360,173
361,170
362,168
363,165
364,162
365,160
366,157
367,156
368,154
369,152
370,149
371,147
372,145
373,143
374,141
375,140
376,138
377,136
378,134
379,133
380,131
381,129
382,128
383,127
384,125
385,124
386,123
387,121
388,120
389,120
390,119
391,118
392,118
393,117
394,115
395,114
396,113
397,112
398,111
399,111
400,110
401,109
402,108
403,107
404,106
405,105
406,104
407,103
408,102
409,102
410,101
411,101
412,100
413,99
414,98
415,98
416,97
417,97
418,96
419,95
420,95
421,94
422,94
423,93
424,93
425,92
426,92
427,91
428,90
429,90
430,89
431,89
432,88
433,88
434,87
435,87
436,86
437,86
438,86
439,85
440,85
441,85
442,85
443,84
444,84
445,84
446,84
447,83
448,83
449,83
450,82
451,82
452,82
453,82
454,82
455,82
456,83
457,82
458,82
459,82
460,82
461,82
462,81
463,81
464,80
465,79
466,79
467,79
468,78
469,78
470,77
471,77
472,77
473,76
474,75
475,75
476,75
477,74
478,74
479,75
480,75
481,74
482,74
483,73
484,73
485,72
486,72
487,72
488,71
489,70
490,70
491,69
492,69
493,69
494,69
495,68
496,68
497,68
498,67
499,67
500,67
501,66
502,66
503,66
504,66
505,65
506,66
507,66
508,65
509,66
510,65
511,65
512,65
513,65
514,65
515,65
516,64
517,64
518,64
519,64
520,64
521,64
522,65
523,64
524,65
525,65
526,65
527,67
528,67
529,67
530,67
531,67
532,67
533,65
534,63
535,61
536,60
537,59
538,57
539,56
540,55
541,54
542,52
543,51
544,50
545,50
546,49
547,48
548,47
549,46
550,46
551,45
552,44
553,44
554,43
555,43
556,42
557,42
558,42
559,41
560,40
561,40
562,40
563,39
564,39
565,38
566,38
567,37
568,37
569,37
570,37
571,37
572,36
573,36
574,36
575,35
576,35
577,35
578,34
579,34
580,34
581,34
582,33
583,33
584,32
585,32
586,32
587,32
588,31
589,31
590,31
591,30
592,30
593,30
594,30
595,30
596,30
597,30
598,28
599,28
600,28
601,28
602,28
603,28
604,28
605,28
606,28
607,28
608,28

Java Application (StripChart View)



Android Application (Main View)



Circuit Diagram