|  |
| --- |
| http://www.intersectionconsulting.com/wp-content/uploads/2011/03/UBC-logo.png  University of British Columbia Faculty of Applied Science Department of Electrical and Computer Engineering EECE 281 – Design Project I |
| DESIGN AND DEVELOPMENT OF |
| AUTOMATED REFLOW OVEN CONTROLLER |
|  |
|  |
|  |

|  |  |  |
| --- | --- | --- |
| Date of Submission: | Monday, February 24, 2014 | |
| Submitted to: | Dr. Jesús Calviño-Fraga, Ph.D. | |
| Submitted by: | GROUP BB  Derek Chan  Nina Dacanay  Aleksander Dordzijev  Glyn Han  Jessica Hua  Kyujin Park | 12345678  21848122  12345678  21847124  12345678  24898132 |

**Table of Contents**

[1.0 Introduction 1](#_Toc380932288)

[1.1 Specifications 1](#_Toc380932289)

[1.1.1 Software Specifications List of files & Brief Summary of each file function: 2](#_Toc380932290)

[1.1.2 Hardware Specifications 4](#_Toc380932291)

[2.0 Investigation 6](#_Toc380932292)

[2.1 Idea Generation 6](#_Toc380932293)

[2.2 Investigation Design 6](#_Toc380932294)

[2.3 Data Collection 7](#_Toc380932295)

[2.4 Data Synthesis & Analysis of Results 8](#_Toc380932296)

[3.0 Design 9](#_Toc380932297)

[3.1 Use of Process 9](#_Toc380932298)

[3.2 Need and Constraint Identification 10](#_Toc380932299)

[3.3 Problem Specification 10](#_Toc380932300)

[3.4 Solution Generation 11](#_Toc380932301)

[3.5 Solution Evaluation 11](#_Toc380932302)

[3.6 Detailed Design 11](#_Toc380932303)

[3.7 Solution Assessment 11](#_Toc380932304)

[4.0 Life – Long Learning 11](#_Toc380932305)

[4.1 Software 11](#_Toc380932306)

[4.2 Hardware 12](#_Toc380932307)

[4.3 Management 12](#_Toc380932308)

[References 13](#_Toc380932309)

[Bibliography 14](#_Toc380932310)

[Appendix iii](#_Toc380932311)

# 1.0 Introduction

Reflow soldering is a common procedure utilized to attach surface mount devices (SMDs) into printed circuit boards (PCBs). Solder paste, a mixture of solder flux and pellets, temporarily secures the components into the board, which is then placed in the oven to melt. This method requires a strict control of time and temperature, allowing the solder to melt and permanently assemble components in great detail. The objective of this project is to design, build, program, and test a system controller that controls the process of reflow soldering.

## 1.1 Specifications

There are 4 selectable reflow profile parameters included: soak temperature, soak time, reflow temperature, and reflow time. These parameters can either be selected from preset values or can be manually adjusted with switches SW8-SW0 and verified by the pushbutton KEY1. Typically in a reflow soldering process, there are 5 reflow states and the profile parameters control these states. These states include: ramp to soak (1-3°C/sec), preheat/soak (60-120 seconds at ~150°C ± 20°C), ramp to peak(1-3°C/sec), reflow (~45-75sec past 217°), and cooling (2-4°C/sec). HEX7-HEX4 displays the total time while HEX3-HEX0 displays the elapsed time for each state. All device to user interactions are displayed through a 2x16 serial LCD display module.

### 1.1.1 Software Specifications List of files & Brief Summary of each file function:

|  |  |
| --- | --- |
| **main.asm** | State-machine |
| **SSR.asm** | Controls the SSR Relay box:  Switch On: SSR enables  Switch Off: SSR disables |
| **Serial\_Port.asm** | 1. Takes and Reads the temperature from the K-Type thermocouple   connected to the MCP3004   1. Displays the temperature to a Python Strip Chart through the Serial Port |
| **ADC Converter** | Analog to digitalized form |
| **Buzzer.asm** | Interface for the Buzzer component |
| **Thermo2.asm** | Loads the variable Temperature\_Measured with the correct temperature in the oven, calibrated with the cold junction outside of the box. |
| **User\_Interface.asm** | 1. Settings Initialization (Settings\_Initialization) called at the beginning of the code, this is the user interface function used to get the temperature settings and stores them into registers.    1. Welcome\_message    2. Soak\_Temperature\_Input    3. Soak\_Time\_Input    4. Reflow\_Temperature\_Input    5. Reflow\_Time\_Input 2. Checking for other inputs (Check\_Inputs)Provides a check to the other user inputs used during the heating process (i.e. force stop / oven open) 3. Status Display / Value Display (Display\_board)Ability to write to the Hex Display / LCD Display and output set messages onto the LCD Display |
| **Door.asm** | Controls the SSR Relay box:  1 if door is open, SSR disable,  0 if door is closed, SSR enable |
| **LCD\_Display.asm** | Write messages to the LCD according to state |
| **Read\_sw5.asm** | Control switches |
| **Thermo\_Python** | Display graph chart on the computer |

Figure : Software Block Diagram

Thermocouple Temperature

(Thermo2.asm)

(

Serial Port

(serial\_port.asm)

(

Interface

(User\_Interface.asm)

(

Toaster Oven

ADC Converter

(adc\_converter.asm)

(

LM335

Room Temperature

Power Control

(SSR.asm)

(

LCD Display

(LCD\_Display.asm)

(

Check Door

(Door.asm)

(

Control Switches

(Read\_sw5.asm)

(

Temperature Chart

(Thermo\_python)

(

Display strip chart

on the computer

State-Machine

(main.asm)

### 1.1.2 Hardware Specifications

Our reflow oven controller is capable of measuring reflow oven temperatures between 0°C and 280°C. In order to measure the temperature inside the oven, a K-type thermocouple with cold junction compensation is used. The thermocouple outputs 41µV/°C. To amplify the excessively small output, it is connected to an OP07 OP-AMP. The resistors R1 and R2, which we used to wire the difference amplifier, are 20KΩ and 47Ω respectively (Talk about how we chose these to calculate our temperature range). The amplifier is then connected to (Talk about how the SSR & PULSE WIDTH MODULATION WORKS.ASK OTHER PEOPLE). The analog signals obtained through the LM355 temperature sensor and from the thermocouple wire are processed through an MCP3004 ADC (analog to digital converter). An NPN 2222A BJT is used to distribute a controlled signal of 5V to the SSR box from the pulse width modulation of the DE2 board. A serial port is also a constant communication tool for the DE2 board to the user’s computer, mapping out a strip chart of the temperature and time.

Figure : Hardware Block Diagram

ICL 7660

Produces -5V

Altera DE2

Collect Data from Channel 0 and 1

MP3004

Analog data into digitalized form

OP 707

Amplifies the reading input voltage

PN 2222A

Transistor, helps to control SSR Box

K-Type Thermocouple Wire

LM335

OVEN

SSR Box

Provide power

Connected to Channel 1

Connected to Channel 0

Produce 5V Input

# 2.0 Investigation

## 2.1 Idea Generation

In order to efficiently generate ideas and a working hypothesis, our group maintained a schedule and an organized list of assignments. Before our first formal meeting, we agreed to read all the files and watch all the videos regarding the project. By doing this, we were informed of the project objectives and acquired time to formulate ideas and a working hypothesis of how the reflow soldering oven should work. In our group meetings, these ideas and hypotheses were discussed and written down for reference later on. We decided that each idea and hypothesis should undergo a performance test and analysis every time the segment in which it is used for is completed, in order to keep track of its function and operation.

During the first meeting, everyone volunteered their area of strength and interest whether it was hardware, software, organization, etc. We deliberately divided the whole project and assigned individual tasks according to these areas of strength and interest. Through this method, each team member is focused on what they are capable of doing, which is appropriately time-efficient in a time-constrained schedule. This was the most effective way to generate ideas because each person is focused on one particular task instead of stressing on the big picture. On the other hand, anyone was welcome to contribute ideas and suggestions to another member’s assigned task to keep the project open-ended.

## 2.2 Investigation Design

The design of our reflow oven controller was established through careful research of the specific component data sheets and examination of the project files and lecture slides provided in the UBC connect website. Individually, we gathered information for our own tasks and later discussed our plan with the group. Every member’s plan of action is recorded into a file that was stored in an online tool called GitHub (please read section 2.3 for further details), which allows access to all the group members for reference. Once the group has heard and approved of the proposal, we continued to work on the design until completion. As aforementioned, each design underwent a performance test and an analysis whenever it is completed in order to keep track of the design’s function and operation. During the performance test and analysis, each member of our group was present to observe and allow individual suggestions and constructive criticism for the design to improve on. Again, these are all recorded and stored into GitHub.

The division of individual tasks definitely made information and data gathering for specific topics more effective. Specifically, Derek Chan was in charge of code organization, unification, and the state machine; Nina Dacanay was mainly in charge of the software coding for the temperature sensor, the serial port, and the timers; Jessica Hua and Aleksander Dordzijev were in charge of the software coding for the user interface; while Glyn Han and Kyujin Park were mainly in charge of the hardware assembly. However, even though we were assigned specific parts, each member of the team was exposed to both software and hardware through soldering components onto the PCB and understanding all the codes used for the main program.

## 2.3 Data Collection

Our group utilized plenty of resources to keep our project consistent with whatever we required to update it with and other tools are used to check the circuitry and debug a code. Such resources include: Github, Facebook, continuity buzzer, voltmeter, and various debugging tools that have been provided. Github is an online hosting code repository service specifically for software development. [1] Our group utilized this popular programming tool to keep our newly written and old codes from mixing together on this cloud based storage. This maintains all our code organized and clean from any data errors while transferring and preserves history of added code and deleted code. Facebook, the popular social networking site is where our group has created a separate page where all our group members can see the latest updates on progress, next scheduled working sessions, and any type of issues that may arise from the project.

We used the voltmeter and the continuity buzzer to check the circuit every time we have soldered on specific sections of the circuit. By doing so this allows us to debug the circuit carefully by identifying issues before the entire circuit is assembled. Without testing the connections and the right voltages at certain locations it may cause room for simple errors that would take hours to identify and resolve, by doing it this way is also very time efficient. The debugging tools provided to us for use are CrossIDE and Spyder. CrossIDE is a debugging program developed by our Professor, it debugs our 8051 assembly code. Spyder is an open source IDE (integrated development environment) that was provided to us and we utilize it to debug our python codes for the serial port interfaces in order to display and print out our strip chart of the temperature and the time. [2]

## 2.4 Data Synthesis & Analysis of Results

Our group verified the validity of our results by testing the circuits and programs with their theoretical and assumed function. For software, we constantly debugged and verified that the function of the program was working the way we had designed it. Verifying the individual parts of the code made it much simpler to identify the location of the malfunction. The validity of the programs we designed were all based on class knowledge and past lab experiences.

As for the hardware and circuitry, their validity was tested through the knowledge we have obtained through our circuits I course. We tested the circuits at specific locations and we compared the physical results we obtained with our theoretical results that we had calculated to verify the validity of the different sections of the project. The validity of the programs and circuits were also tested by combining smaller parts of the program with the hardware. This allowed verification for the larger portions of the project with abovementioned techniques. Additionally for the end result, when we ran through the entire reflow soldering states, we compared it to the ideal theoretical temperatures and times. After the project was running the way we had planned and designed, we tested it with a PCB and we compared our result with online results and videos that were available to us.

# 3.0 Design

## 3.1 Use of Process

In the early design stages of the project, it was quickly evident that the complete oven controller would require many basic components interacting in a complex manner. Our group recognized the complexity in having many interacting parts and decided to simplify the project through modulation. This meant that we identified each feature of the oven controller and made it into its own module with detailed specifications. We planned to have clear specifications for each module so that when we finally needed to put all the components together, we would be able to easily troubleshoot issues and integrate the features of our modules. Each module would be abstracted so that only two or three functions would need to be used, and the complexity in implementation would be hidden while we combined all of our components together.

## 3.2 Need and Constraint Identification

Each feature that was required in the project was identified as an individual need. The needs we identified were:

1. ability to generate tones
2. ability to determine if the door was open or closed
3. ability to display text messages for the user
4. ability to transmit serial data to a computer
5. ability to turn the oven on and off using a solid state relay
6. ability to measure temperature in the oven
7. ability to measure time and display time to the user
8. ability to handle user inputs to modify parameters

## 3.3 Problem Specification

From the needs we identified, we created modules that met each need, each with their own specifications. The following list of modules are numbered respective to the needs listed above:

1. Buzzer
   * Generate a continuous tone
   * Generate multiple short tones
2. Door
   * Update a variable according to whether the door is open or closed
3. LCD
   * Write and display specific messages to the user
4. Serial Port
   * Transmit the current temperature through the serial port
5. SSR (Solid State Relay)
   * Enable and disable the oven attached to the SSR
6. Thermocouple Input
   * Measure the temperature in the oven
     + Determine the voltage generated by the thermocouple wire
     + Convert the voltage into a temperature value
     + Determine the ambient temperature (cold junction temperature)
     + Take a difference between the inside and ambient temperatures
     + Store the temperature as a variable
7. Timer
   * Keep a total time for the entire soldiering process
   * Keep a separate process time for separate processes
8. User Interface
   * Take in user 3-digit values for soak and reflow temperatures, and soak and reflow durations
   * Display back to the users their desired values and times and allow confirmation of values

## 3.4 Solution Generation

## 3.5 Solution Evaluation

## 3.6 Detailed Design

Each individual component was required to provide a subroutine for initialization and for each specified feature. Please refer to Appendix B for each header file for our components, and Appendix C for the source code.

## 3.7 Solution Assessment

# 4.0 Life – Long Learning

## 4.1 Software

The most useful course was undoubtingly EECE 259 where we learned everything about microcomputers knowledge that we needed for this project.

We spent most of our time and effort into integrating all the functions into one state-machine. The challenge was lack of experience and knowledge for managing complex software systems.

In order to overcome the problem, we learned to use GitHub where we could easily keep track of everyone’s work. We created a header file for each code file, defining its purpose, and its use of variables and registers. Additionally, we referenced the lecture notes which describe to write codes for a state machine. From this, we became more efficient programmers.

## 4.2 Hardware

EECE 251 was very helpful because we gained an experience with soldering chips on the circuit board and we learned how to check the connection of soldered iron by using the voltmeter. Moreover, for EECE 251 labs, we already designed and built an amplifier and a BJT circuit which were the major part of the hardware.

## 4.3 Management

The group consisted of six people and each member’s task was very technical. In order to succeed as a group, we spent time creating a schedule and dividing our workload appropriately. Our group focus was time management. We knew in order to finish the project it would require a lot of our time but at the same time, we had other courses to keep up with. Also, we wanted to finish the project at least three days before the deadline. After we were aware of what our goals were, it was easy to be committed in order to reach our goal. We learned to manage our task appropriately to our conditions.

# References

|  |  |
| --- | --- |
| [1] | K. Finley, "Github Has Surpassed Sourceforge and Google Code in Popularity - ReadWrite," Say Media Inc., 2 June 2011. [Online]. Available: http://readwrite.com/2011/06/02/github-has-passed-sourceforge#awesm=~owugMKdmHVVJVl. [Accessed 20 February 2014]. |
| [2] | R. Grehan, "Spyder Review - Techworld.com," International Data Group, 8 September 2010. [Online]. Available: http://review.techworld.com/applications/3238833/spyder-review/. [Accessed 20 February 2014]. |

# Bibliography

|  |  |
| --- | --- |
|  | H. Han - Way, Using the MCS - 51 Microcontroller, New York: Oxford University Press, Inc., 2000. |

# Appendix