Reflow Oven Controller

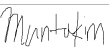
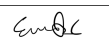
Team A9

March 1st, 2024

Course: ELEC 291/292 Electrical Engineering Design Studio I

Section: L2A

Instructor: Dr. Jesus Calvino-Fraga

Student #	Student Name	%Points	Signature
67286750	Tomaz Zlindra	-	
21475330	Martin Lieu	-	
7106221	Muntakim Rahman	-	
76029933	James Huang	-	
17542499	Emile Jansen	-	
46634911	Kyden McCaskill	-	

Contents

1 Introduction

1.1 Objective

The objective of this project was to design, program, and test a reflow oven controller whose purpose is to be able to safely and reliably assemble surface mounted devices onto printed circuit boards. The design of this project is built upon the concepts explored in Labs 1-3, including LCD interfaces, push buttons, timers, ADC converters for reading external inputs such as temperature, and a serial communication interface. In addition to these concepts, voltage amplifiers and PWM peripherals were also used to meet our design needs.

The intended outcome of this project was to leverage the limited capabilities of the N76E003 microcontroller[7] to solder EFM8 boards for the second half of the course. An image of the breadboard used to design our controller can be found in the appendices figure 11.

1.2 Specifications

The tables below contain the hardware and software specifications of this project.

Circuit	Chips	Other Components
Main	N6E003 Micro-Controller BO230XS USB adapter	N/A
Voltage Amplifier	OP07CP Op-Amp LMC7660 Voltage Converter	2x 10MΩresistors 2x 33kΩresistors 2x 10uf capacitors 1x potentiometer
Temperature Sensing	LM335 Temperature Sensor	K - Thermocouple 1x potentiometer 2kΩresistor
PWM	N/A	2N3904 Transistor 1x 330Ωresistor
Multiplexer	N/A	5x push buttons 5x 1N4148 diodes

Table 1: Hardware Specifications

Assembly	Python
UI on LCD Settable Soak time and temperature Settable Rise time and temperature Temperature calculations from ADC Transmit Temperature to Serial Transmit State to Serial and UI	Strip chart of Temperature Data Date Time Identified CSV Data CSV File Cloud Upload

Table 2: Software Specifications

1.3 Block Diagram

The figure below contains a block diagram showing the flow of information for both hardware and software in our design. It includes the controller made on the N76E003 as well as the interface controlled by the user and the major hardware components.

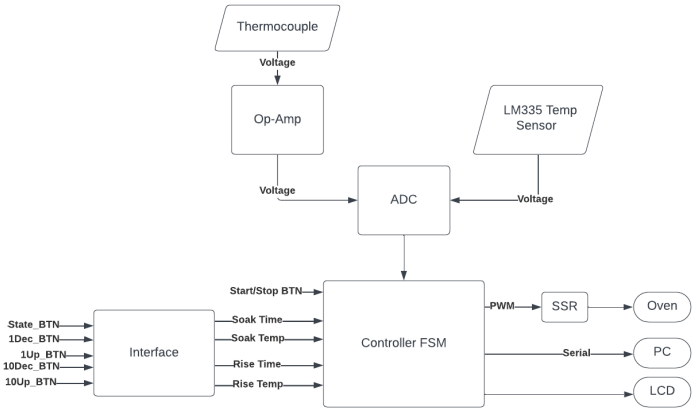


Figure 1: Block Diagram

2 Investigation

2.1 Idea Generation

Our idea generation process began with a thorough discussion of the individual requirements of a fully functional controller. We then dissected the problem into smaller, manageable parts, enabling effective task allocation for firmware and hardware design. Initially, we developed individual components in isolated test environments to confirm details, such as the functionality of the ADC and PWM peripherals, the performance of the op-amp circuit, and the intended state changes. Integrating these functional components into a single, cohesive product was the final step, allowing us to fine-tune initial designs collaboratively through end-to-end testing.

2.2 Investigation Design

In the investigation design phase, we utilized circuit simulation programs like LTspice to test the feasibility of hardware components such as the op-amp prior to manual construction. LTspice provided a platform for designing, altering, and simulating the op-amp values to meet specifications. Additionally, we archived our past experiments on a Google Cloud server, enabling remote access to previous experimental data. This ensured that team members could generate graphs and analyze data, maintaining up-to-date knowledge.

2.3 Data Collection

The ADC and temperature data collection and analysis were conducted through a three-way analytical approach using the oven. Firstly, the "kconvert" script facilitated a comparative analysis between numerical readings and actual temperatures, allowing for precise calibration. Subsequently, a custom-designed strip chart application illustrated the data, facilitating the assessment of con-

sistency against expected patterns. Lastly, upon completion of the task, all temperature readings were uploaded and logged onto a Google Cloud server, providing a record of timestamps and temperatures for future reference and analysis[3].

Furthermore, the utilization of instruments such as an oscilloscope facilitated the examination of PWM implementation behavior, guaranteeing adherence to specifications. After verifying the desired functionality, we integrated the signal into our finite state machine, allowing joint observation across states to ensure continued compliance with specifications.

2.4 Data Synthesis

Initial data was obtained through trial runs. It was imperative that appropriate temperatures were reached or maintained during and throughout critical stages such as state transitions or holding periods. To align graphical representations with desired diagrams, modifications to the PWM signal were made to strengthen or weaken power output, ensuring graphical congruence. Once power optimization met specifications, further data synthesis proceeded.

2.5 Analysis of Results

In analyzing the results, we validated our measurements and conclusions to the degree of error and limitations illustrated by example test programs and charts. This process involved scrutinizing data for precision and accuracy while considering discrepancies in trial runs.

3 Design

3.1 Use of Process

To understand the complex challenges that needed to be addressed, we created a data flow diagram of our system to understand the state changes as we progressed through both successful and unsuccessful reflow cycles. This was a key step in understanding the requirements of the project and what we must consider in our solution.

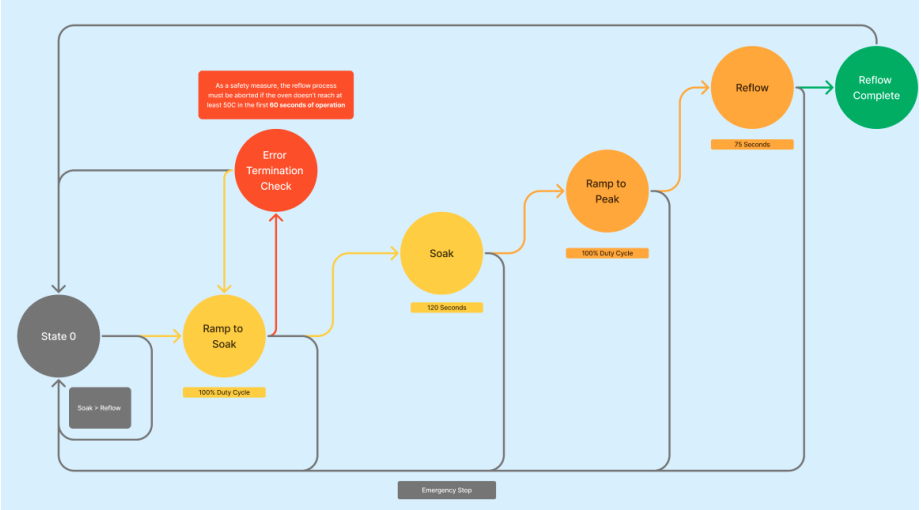


Figure 2: Finite State Machine Diagram

This diagram allowed our group to determine the conditions that changed the state and the code that needed to be done once versus iteratively.

3.2 Need and Constraint Identification

To identify needs for a functional oven controller, we listed the functionalities expected of a fully functional Reflow Oven Controller. The main requirements for this project include the following:

- Must be written in 8051 Assembly.

- Must measure temperatures $25°\text{C} \leq T \leq 250°\text{C}$, accurate to within 3°C.
- Must include editable soak and reflow temperatures.
- Must include safety features, in particular automatic cycle termination on error.
- Must include button functionality for initiating and terminating the reflow process.
- Must include an LCD Display with configuration and current temperature readings.
- Must communicate with a Software UI via serial communication and transmit temperature data.

Note: These were listed in a requirement PDF provided by Dr. Jesus Calvino-Fraga[10].

In addition to these, we also identified constraints that would affect the design of the controller. These include:

- The N76E003 micro controller has a limited number of pins, timers, and peripherals.
- Assembly language is not as intuitive as higher-level languages and can easily become complex in collaboration.

To get around these, we tried to start off with simplified functionality and iterate on it to progress towards meeting project requirements. We collaborated via a Github repository for source control and keeping track of iterations. This allowed us to move fast on functionality, such as the finite state machine, and then iterate on it to add more features, such as the temperature sensor and the serial communication interface.

3.3 Problem Specification

It's become evident throughout this project that an initial listing of problems is a strong foundation for development. However, often we have found that there are hidden problems in feature development. In summary, the individual functionality of hardware and firmware modules did not guarantee smooth integration. Collaboratively stepping through the process of reflow soldering with our prototype ensured we addressed key points, including the planned state change and cycle termination functionality, the intended conversion of ADC readings from the K-type thermocouple voltage, and the consideration of arithmetic operations between temperature variables.

3.4 Solution Generation

We found that the best way to develop our solution was through a four-step process.

1. Discussing the individual requirements and assigning development to team members.
2. Developing sub modules in a test environment.
3. Integrating the sub modules into a single, cohesive product.
4. Exhaustively test and debug functionality until requirements are met.

Tackling the problems in the section above in a collaborative manner allowed us to address them comprehensively and was a direct contributor to our final product. We also found that ownership over individual subcomponents allowed us to have team members who best understood particular behaviors. When we encountered problems, we raised concerns and asked questions to move towards fixing and integrating them.

One issue that arose during development was the need to have distinct ADC functionality for the K-type thermocouple and the LM335 temperature sensor[4]. We initially tried to have identical ADC functionality for both, but this was not feasible due to the different voltage ranges and the need for additional conversion of the thermocouple voltage to temperature. We resolved this through incremental changes and testing on our LCD. In order to ensure our firmware wasn't a black box, we printed debugging information on the LCD. This was a makeshift solution to being unable to halt the microcontroller and step through the code in a debugger.

3.5 Safety/Professionalism

To ensure our product was safe, we placed emphasis on accurate temperature reading and transmission. Once we had a semi-functional ADC module, we tested the calculated temperature with the multimeter program provided by Dr. Jesus Calvino-Fraga[10]. This allowed us to ensure that the temperature readings were accurate to within 3°C. In addition, the cycle termination functionality was key to halting a faulty reflow process. This both allowed us to turn off the oven and lower temperatures, as well as prevent damage to our EFM8 boards. Furthermore, we added additional functionality to prevent the configuration of reflow temperature below the soak temperature. This feedback was displayed to the user on the LCD. To ensure professionalism, we maintained a clean and organized codebase, and we also maintained a professional attitude in our communication and collaboration.

3.6 Detailed Design

3.6.1 Firmware Design

Our final state machine helped us integrate submodules for the PWM, ADC, and serial transmission. In the figure below, it is noticed that our firmware

peripherals send datastreams directly to the serial port and the LCD. For arithmetic operations in our Math modules, we use hexadecimal variables, and this is sent to our Python software application for data post-processing. The LCD however, accesses a Binary Coded Decimal copy of our calculated temperature. This is to prevent the complications of $bcd2hex$ operations in our arithmetic. We decided it was simpler and a more straightforward solution to have dedicated variables for these separate datastreams.

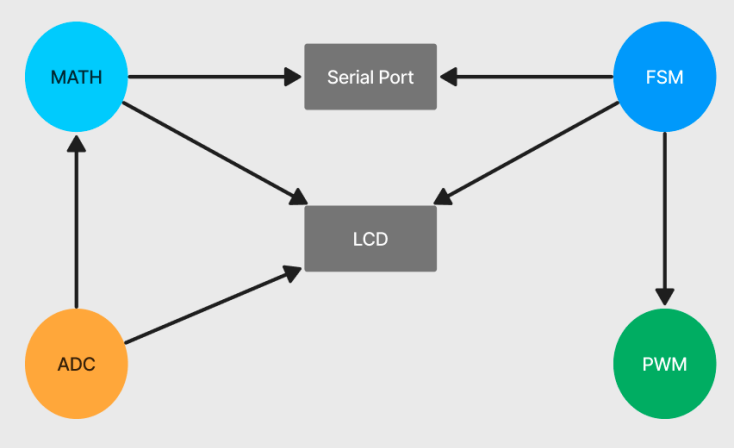


Figure 3: Software Modules Diagram

After discussing the slow response time of the oven, we decided to implement a $1s$ PWM period.

The ADC was set to store data in 4-byte variables. These were offset to ensure we added the correct digits from the thermocouple temperature and the LM335 temperature sensor.

```
Get_THERMOCOUPLE_TEMP:
  ; Read AIN7 on Pin 14
  ANL ADCCON0, #0XF0
  ORL ADCCON0, #0X07 ; Select ADC Channel 7
  LCALL Read_ADC_Avg
  LCALL Convert_Voltage
Calculate_THERMOCOUPLE_TEMP:
```

```
  LCALL hex2bcd
  MOV Output_Voltage+3, BCD+3
  MOV Output_Voltage+2, BCD+2
  MOV Output_Voltage+1, BCD+1
  MOV Output_Voltage+0, BCD+0

  Load_y(1000) ; Vout * 1000 / 12424
  LCALL mul32
  Load_y(12424)
  LCALL div32
  Load_y(100)
  LCALL mul32

  ; Save Result to Use Later.
  MOV THERMOCOUPLE_TEMP+3, X+3
  MOV THERMOCOUPLE_TEMP+2, X+2
  MOV THERMOCOUPLE_TEMP+1, X+1
  MOV THERMOCOUPLE_TEMP+0, X+0

  RET
```

As mentioned previously, we store a BCD copy of our calculations for the LCD. This is stored in 2 bytes, as shown below.

```
Write_Oven_BCD:
  MOV X+0, OVEN_TEMP
  MOV X+1, #0
  MOV X+2, #0
  MOV X+3, #0

  LCALL hex2BCD
  MOV OVEN_BCD+1, BCD+1
  MOV OVEN_BCD+0, BCD+0

  RET
```

Our calculated oven temperature was stored in 1 byte. This was due to our

interest in integer values between 25°C and 250°C.

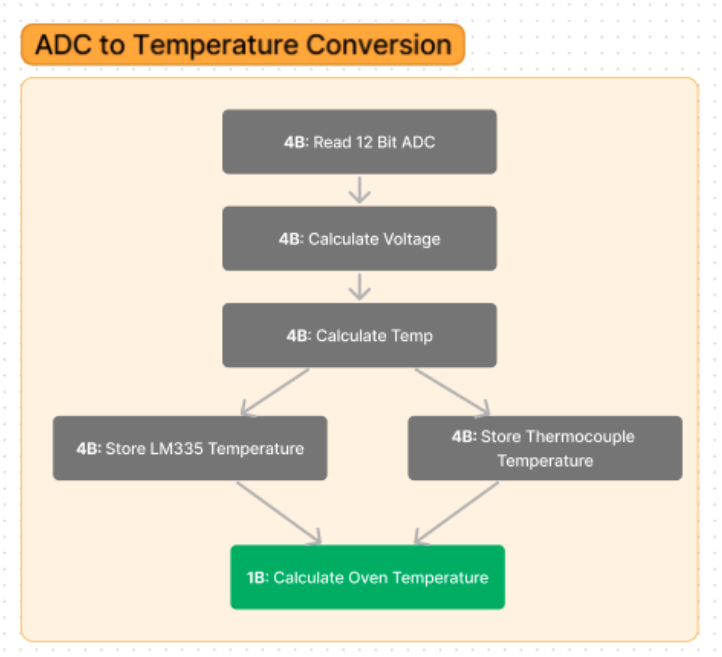


Figure 4: ADC Dataflow Diagram

To compare the current oven temperature with the 1 byte reflow and soak temperatures, we used the 8051 assembly compare instruction. This allowed us to set flags for the state machine to progress through the reflow cycle.

```
Below_Temp_Flag: DBIT 1 ; For State Changes
Error_Triggered_Flag: DBIT 1 ; For Cycle Termination
...
Reflow_Soak_Flag:   DBIT 1 ; For State Changes
State_TX_Flag: DBIT 1 ; For Serial Transmission of State Number
```

We applied the same flag logic to the serial transmission of the state number. This allowed us to transmit the state number to the software UI and to the LCD. Doing this enabled us to overcome the problem of having a datastream dominated by state numbers rather than relevant temperature data. Instead, we used this flag during state changes to transmit this to the software UI.

With this design, we are able to achieve data reliability in our data post-processing, as shown in the *csv* data graphed below.

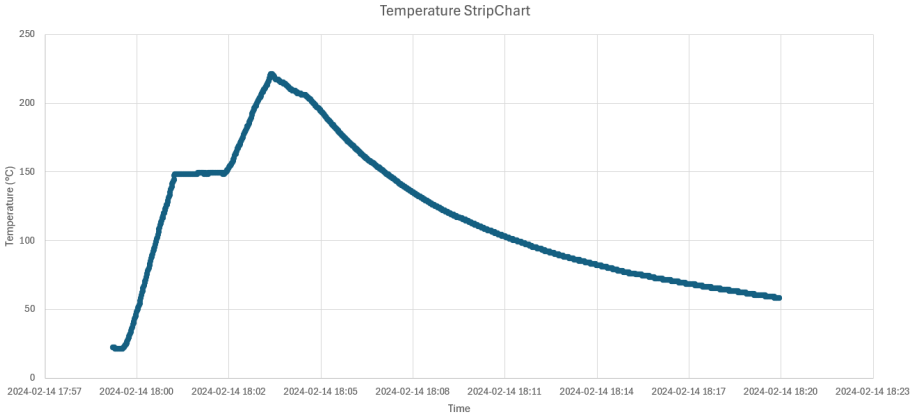


Figure 5: Reflow Data

3.6.2 Voltage Amplifier

The purpose of this section is to highlight the design of a voltage amplifier used to amplify the small voltage difference of a k-thermocouple of $41\frac{\mu V}{C^\circ}$ to a voltage difference of approximately $12.42\frac{mV}{C^\circ}$. This was achieved using the OP07CP[7] Op amp that was included in the Project 1 kit. The circuit schematic of this amplifier was modeled in LTspice and is shown in the figure below.

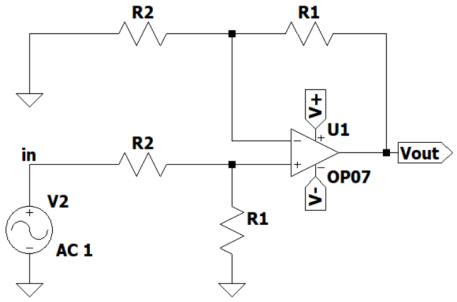


Figure 6: Voltage Amplifier Circuit Diagram

The design of this circuit comes from the course notes provided by Dr.

Calvino-Fraga. In addition to making the circuit schematic, LTspice was used to determine the value of resistors R1 and R2. Through simulation, it was found that the best resistor values from those given in the Project 1 lab kit were 10MΩ for R1 and 33kΩ for R2.

When integrating our voltage amplifier into our design, we had to use the LMC7760[5] voltage converter to provide the appropriate voltage to our Op-Amp. These two chips were then wired to a breadboard to achieve the appropriate voltage amplification. As an addition a potentiometer was connected to the OP07CP to null the voltage offset, this was done to obtain more accurate and consistent readings.

3.7 Solution Assessment

We assessed our design through a set of tests. Initially, before we had a working version of the controller, we would first test the FSM, ADC conversions, PWM, OP Amp circuit and Serial separately (typically by having an assembly file that would simulate it). Once we knew each module was working individually, we integrated them one by one, testing them completely before adding the next module. To validate whether our system worked, we would use electrical tools such as oscilloscopes, multimeters and of course, the LCD display. Most of these systems were debugged using a test-and-try procedure.

Below is a brief overview of how we debugged certain systems:

- FSM. This was debugged by displaying the state number, the timer and the resulting timer counter on the LCD, with a button to change state when in a non-timer state.
- PWM. This was debugged by displaying the wave on the oscilloscope and viewing the shape and period of the square wave.

- ADC. This was debugged by displaying the full thermocouple and room temperatures on the LCD and ensuring the addition is correct.
- OP Amp. This was simply debugged by ensuring proper wiring, simulating on LTSpice, and finally using the multimeter to obtain the voltage readings.

To validate our oven temperature data, we used a python script provided by Dr. Calvino-Fraga, which would directly convert the voltage difference across the thermocouple into temperature. When our design was complete, we let our controller progress through the reflow process, comparing the difference between the temperature sent over serial and the one displayed by the script. After some fine tuning of the op amp amplification calculations and implementing a 0 offset on the op amp, we noticed the temperature difference was never more than 2 degrees celcius. The temperature difference seemed larger than expected due to having only 3 significant figures (0 decimal places) being sent over by the serial port. We were also able to display the temperature on the python graph to see the live temperature as shown in Figure 5. This allowed us to modify the duty cycle of the PWM signal to ensure the reflow and soak states remain approximately the right temperature.

Our design has many strengths and weaknesses. The strengths include:

- Reliable data. The data shown by the serial port was always identical to the one displayed on the LCD, and never spiked due to secure hardware connections.
- Deterministic state transitions. We studied the state transitions extensively with plenty of individual testing early in the design process.

- Safety features, including the cycle termination functionality and the configuration handling of the soak and reflow temperatures.

Unfortunately, this design also has many weaknesses, such as:

- Used excessive breadboard space. The controller was placed on two breadboards, which requires more connections and visually appears more bulky.
- Ameliorated temperature accuracy. Although our data was approximately 1-2 degrees celsius off from the real temperature, this should've been more accurate.
- Codebase requires refactoring to be more transparent. Submodules can appear as a black box if accessed by new individuals (e.g. ADC arithmetic). Variable names require more clarity and should follow standard naming convention. Codebase maintainability was benefited from migrating most submodule logic to dedicated *.inc* files. This should be continued to ensure the $fsm.asm$ file includes only high level logic (i.e. only include function calls).

We believe, with more emphasis on these issues, these weaknesses could've been addressed.

4 Life Long Learning

Through this project, we gained valuable experience that would help us grow in our respective fields of engineering. We initially encountered challenges when using assembly language to create the basic features, due to our relative lack of experience with it. We also encountered challenges with signal data processing and attempting to send and receive data through the serial port.

On the hardware side of things, we had to learn how to assemble op-amp circuits and multiplex systems that would be important to our final product. We were

able to use and expand upon the programming knowledge we learned in ELEC 291, CPSC 259, and CPEN 211 to program the necessary modules needed to meet the requirements. Through collaboration, cooperation, and communication, we were able to integrate multiple hardware and firmware modules into a single, cohesive product. Utilizing the oscilloscope and multimeter, we were able to successfully perform signal processing and validate our design.

Additionally, we were able to delve into data post-processing and object-oriented software development in our Python applications. We applied the networking principles from ELEC 331 as well in our automation with the cloud server.

5 Conclusions

By the end of the project, we were able to develop a fully functional reflow oven controller. With it, we could control the temperature of the oven to use the reflow soldering method to assemble the EFM8 boards needed for Labs 4 and 5. In addition, we explored additional engineering principles in our feature development. This controller will serve as a strong foundation for future projects and was a great learning experience for our team.

6 References

[1] Calvino-Fraga & University of British Columbia. (n.d.). *ELEC291_Microcontroller_KIT_2023_32*. University of British Columbia. [Online]. Available: https://people.ece.ubc.ca/eng-services/files/courses/ELEC291_Microcontroller_kit_2023_32.pdf

[2] Intel. (n.d.). *MCS 51 MICROCONTROLLER FAMILY USER'S MANUAL*. [Online]. Available: https://canvas.ubc.ca/courses/130658/files/30768826?module_item_id=6480024

[3] LM335 data sheet, product information and support — TI.com. (n.d.). [Online]. Available: https://www.ti.com/product/LM335

[4] LMC7660 data sheet, product information and support — TI.com. (n.d.). [Online]. Available: https://www.ti.com/product/LMC7660

[5] NPN General - Purpose Amplifier. (n.d.). In ON Semiconductor. ON Semiconductor. [Online]. Available: https://www.onsemi.com/pdf/datasheet/2n3904-d.pdf

[6] Nuvoton. (2020). Nuvoton 1T 8051-based Microcontroller N76E003 datasheet. [Online]. Available: https://canvas.ubc.ca/courses/130658/files/30769186?module_item_id=6480289

[7] OP07C data sheet, product information and support — TI.com. (n.d.). [Online]. Available: https://www.ti.com/product/OP07C

[8] OpenAI. (n.d.). *ChatGPT*. Retrieved February 22, 2024, from https://chat.openai.com/

[9] The University of British Columbia. (n.d.). [Online]. Available: https://canvas.ubc.ca/courses/130658

Bibliography

[1] Calvino-Fraga & University of British Columbia. (n.d.). *ELEC291_Microcontroller_KIT_2023_32*. University of British Columbia. [Online]. Available: https://people.ece.ubc.ca/eng-services/files/courses/ELEC291_Microcontroller_kit_2023_32.pdf

[2] Intel. (n.d.). *MCS 51 MICROCONTROLLER FAMILY USER'S MANUAL*. [Online]. Available: https://canvas.ubc.ca/courses/130658/files/30768826?module_item_id=6480024

[3] Jie Jenn. (2021, April 4). Using Google Cloud Storage API in Python for beginners [Video]. YouTube. [Online]. Available: https://www.youtube.com/watch?v=pEbL_TT9cHg

[4] LM335 data sheet, product information and support — TI.com. (n.d.). [Online]. Available: https://www.ti.com/product/LM335

[5] LMC7660 data sheet, product information and support — TI.com. (n.d.). [Online]. Available: https://www.ti.com/product/LMC7660

[6] NPN General - Purpose Amplifier. (n.d.). In ON Semiconductor. ON Semiconductor. [Online]. Available: https://www.onsemi.com/pdf/datasheet/2n3904-d.pdf

[7] Nuvoton. (2020). Nuvoton 1T 8051-based Microcontroller N76E003 datasheet. [Online]. Available: https://canvas.ubc.ca/courses/130658/files/30769186?module_item_id=6480289

[8] OP07C data sheet, product information and support — TI.com. (n.d.). [Online]. Available: https://www.ti.com/product/OP07C

7 Appendices

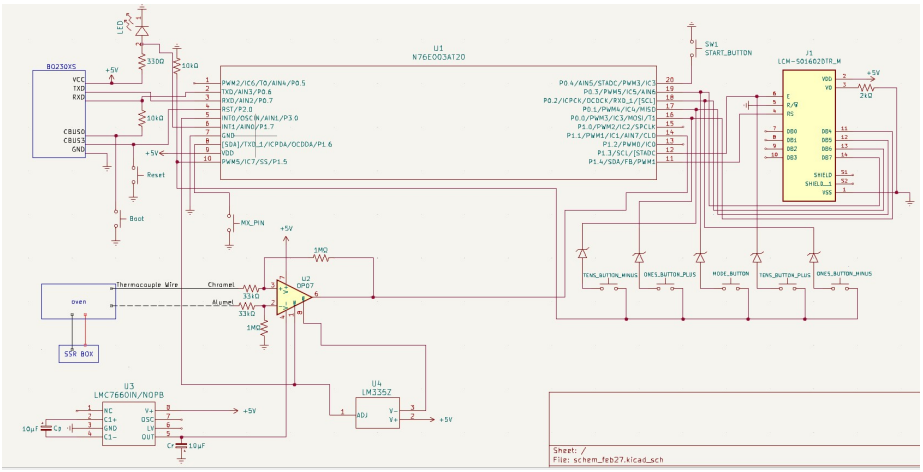


Figure 7: Full Size Schematic of Circuit

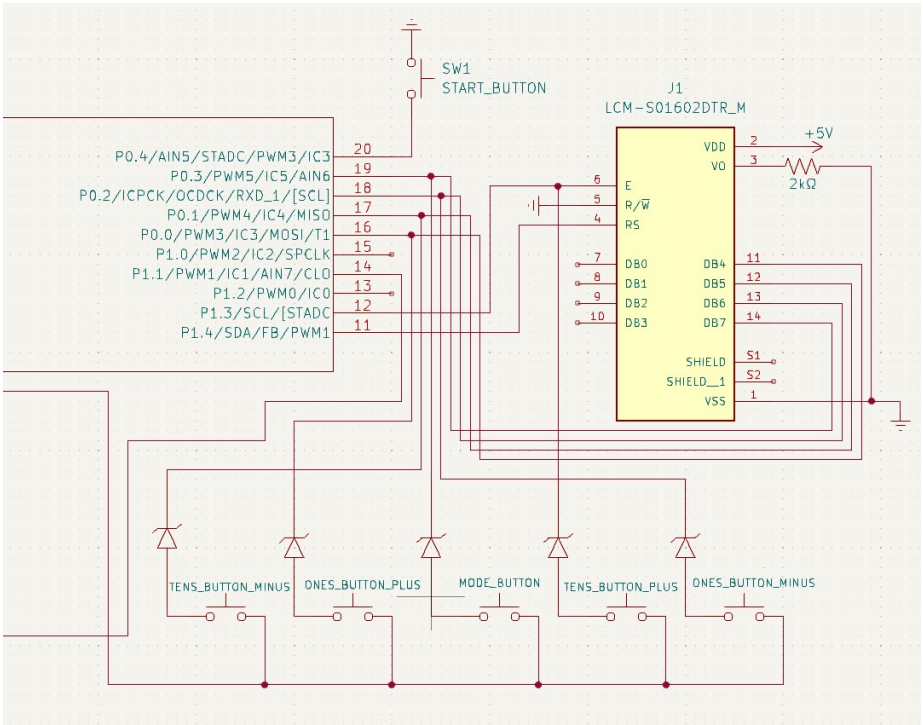


Figure 8: LCD and Button Connections to Microcontroller

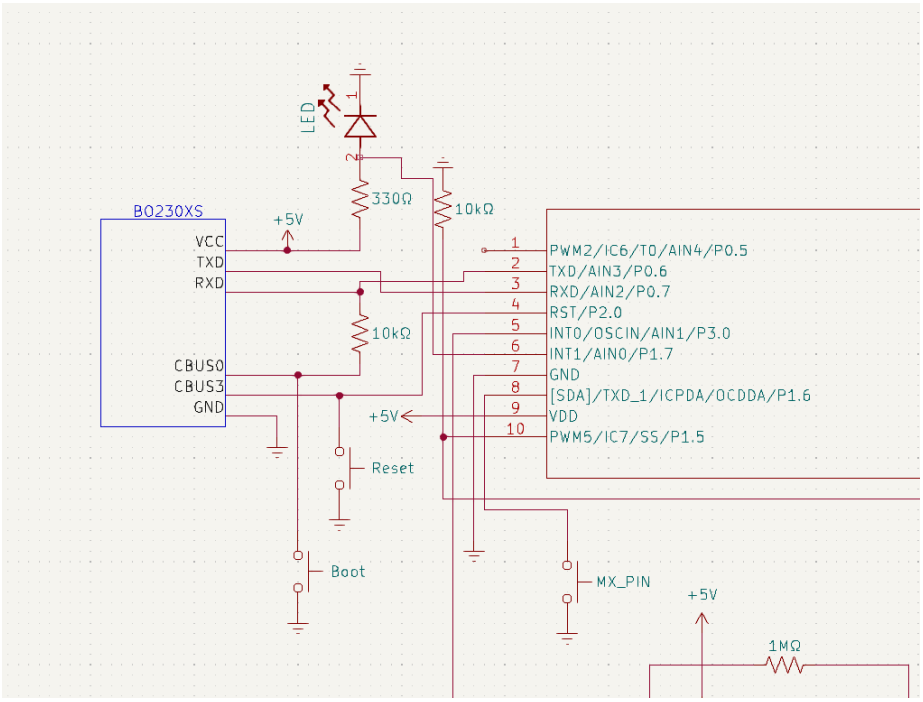


Figure 9: UBC Serial Connector to Microcontroller Pin Connection

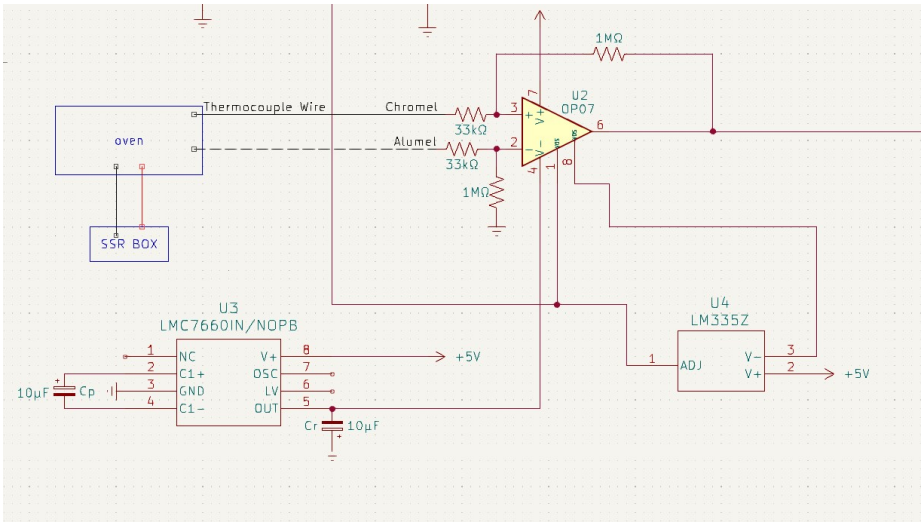


Figure 10: Op Amp and Thermocouple Circuit

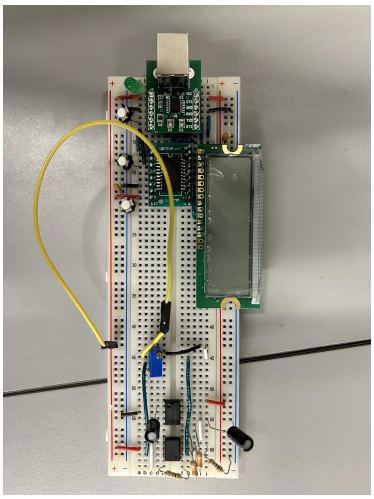


Figure 11: Circuit of Interest