Project Title

DESIGN DOCUMENT

SDDEC01-22
Client: Mark Hansen from Minufacture
Faculty Advisor: Dr. Philip Jones
Rachel Teberg – Team Lead
Stone Widder – Hardware Lead
Charles Sang - Hardware Support
Joshua Baringer - Software Lead
Evan Pasero – Hardware Support
Team Email: sddec01-22@iastate.com
Team Website: sddec22-01.sd.ece.iastate.edu

Revised: Dec 8 2022, Version 2

Executive Summary

Development Standards & Practices Used

- Trade study for display and microcontroller
- Thermocouple Sensor Circuit
- Version control with GitLab
- PCB design using KiCad
- GUI implementation with Python
- Embedded Linux Operating System
- Code of Ethics
- IEEE 802.11-2020
- IEEE/ISO/IEC 29119-2-2021
- IEEE 15939-2008

Summary of Requirements

- Must combine temperature sensors into one control system
- Must track and send data to the cloud
- Must cost 250 dollars or less
- Must have a physical and web UI

Applicable Courses from Iowa State University Curriculum

- EE 201 - Electric Circuits
- EE 230 - Electronic Circuits and Systems
- CPRE 288 - Embedded Systems Introduction
- EE 324 - Signals and Systems 2
- CPRE 488 - Embedded Systems Design
- CPRE 308 - Operating Systems Principles and Practice
- COM S 309 - Software Development Practices
- COM S 252 - Linux Operating System Essentials

New Skills/Knowledge acquired that was not taught in courses

- Project management
- Trade Studies + low level market analysis
- PCB design

Table of Contents

1.Team

1.1 Team Members:

1. Rachel Teberg
2. Joshua Baringer
3. Stone Widder
4. Evan Pasero
5. Charles Sang

1.2 Project Management Style Adopted by the Team:

1. Waterfall

1.3 Project Management Roles:

1. Rachel - Team Lead
2. Stone - Hardware Lead
3. Evan – Hardware Support
4. Joshua - Software Lead
5. Charles - Hardware Support

2. Introduction

2.1 Problem Statement

Our client owns Minufacture which is a small company that sells shredding and injection molding machines dedicated to helping hobbyists interested in sustainability. This allows users to create new items by shredding unwanted plastic, melting it down, and injecting it into mold. The current injection molding machine has two temperature control systems that must be set individually on separate interfaces. This requires the user to know the specific melting temperature of the plastic they are using. Thus, our client needs a new control system that can control both temperatures via one interface, can choose the temperatures based on the type of plastic entered, and can keep the temperature within 10 degrees of the set point.

2.2 Requirements & Constraints

1. Functional requirements
    a. Must work with provided test stand
    b. Must combine temperature sensors into one control system
2. Resource requirements
    a. Must cost 250 dollars or less
3. Qualitative aesthetics requirements
    a. Must fit the current design
4. UI requirements
    a. Must have a single physical UI that can control both sensors

2.3 Engineering Standards

1. IEEE/ISO/IEC 29119-2-2021
    a. This document specifies test processes that can be used to govern, manage and implement software testing for any organization, project or testing activity. It comprises generic test process descriptions that define the software testing processes.
    b. Since we will be using, implementing, testing of software to program the microcontroller and build the PID controller we need to be aware of the use of proprietary software and building generic testing software we need to adhere to safe standards and implementation.
2. IEEE 15939-2008
    a. This International Standard defines a measurement process applicable to system and software engineering and management disciplines. The process is described through a model that defines the activities of the measurement process that are required to adequately specify what measurement information is required, how the measures and analysis results are to be applied, and how to determine if the analysis results are valid.
    b. Our product will be taking temperature measurements from thermocouples and relaying it to microcontroller software while also being able to maintain it thus this standard provides great guidelines when designing such systems as ours to give confidence of a professional standard achieved during production.

2.4 Intended Users and Uses

1. This project is good for hobbyists and environmentalists. As a flexible injection molding machine, it is a perfect solution for people interested in a low-cost way to create miniatures or functional molds. Additionally, it comes with and can use all recycled plastics, making it appealing to those dedicated to environmental causes.
2. Our addition to the Minufacture injection molder will make it more accessible to a wider variety of users, as our display and board will streamline the process of setting temperatures. The temperature setting process will also make the machine more versatile, as it will be able to safely and effectively hold the heating element at temperatures appropriate for a wider variety of plastics.

2.5 Security Concerns

We do not have many security concerns as the device is not normally connected to the internet and will shut off if it gets too hot. However, there is a small period during the initial setup when the device must be connected to Wi-Fi. We are not concerned about this because it is such a small amount of time, and it is going to be downloading from a repository maintained by our client. The only other concern we can think of is that the device will have exposed USB ports. As we were not able to implement touch controls, these ports will be used by keyboard and mouse. Once implemented with the rest of the design the Beagle bone will also be in a project box so the likelihood of something being downloaded to the system via the USB ports is small.

3 Project Plan

3.1 Project Management/Tracking Procedures

We are using the Waterfall project management style. We decided on this style because we felt that the linearity of the style would help us stay on track better. There is a lot of interdependence between the tasks thus the phase structure of the waterfall method fits well with our project. We use a combination of Gitlab and discord to keep track of our progress.

3.2 Task Decomposition

Phase 1

- Project Research
- Basic planning/ setup
- Design hardware block diagram

- Create software specifications document

Phase 2

- Basic prototype using off the shelf components
- Choose hardware components
- Choose software architecture
- Design circuits
- Create server
- Implement PID on Prototype

Phase 3

- Design/print PCB
- Create Physical UI

Phase 4

- Integrate all systems
- Do full system testing

Phase 5

- Implement AWS cloud with GUI

Phase 6

- Final Testing
- Polishing

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

Our project milestones can be broken down into six sections.

1. Research what existing products we can use, and do trade studies on various components in our system
2. Research and gather software libraries to be used to control the PIDs, read from the thermocouple, set up the GUI, and store the data for the profiles
3. Test circuits for the thermocouples, relays, and indicator LEDS
4. Create PCB, implement the GUI and software to control the heating elements
5. Debug problems with our system and make changes to PCB and software
6. Finalize our product
7. Present our product to our client and make sure that we document everything well since the project is going to be open source and will have other teams working on it

We are using the waterfall development process to complete each milestone in sprints. We use our clients' meetings to demo the things that we have worked on during the sprint. We then use his feedback to determine what we need to do in the coming weeks.

3.6 Personnel Effort Requirements

Include a detailed estimate in a table with a textual reference and explanation. This estimate shall be done on a task-by-task basis and should be the projected effort in the total number of person-hours needed to perform the task.

	Josh	Charles	Stone	Rachel	Evan
Board		4/wk			3/wk
Controls	1/wk	2/wk	3/wk	2/wk	1/wk
Software	4/wk		3/wk	2/wk	
Admin	1/wk			2/wk	2/wk

Table 1

Per our milestones we are less interested in specific tasks and more interested in achieving milestones. These milestones are related to the system, and each of these sections (except for admin) relate to part of it. Board relates to the physical aspects of the system, where the EEs spend more time. Controls are also mostly handled by EEs and relate to the PID and low-level software. Software refers to the website, IOT, and cloud connection. Josh will head this, but EEs will help him where he needs it.

3.7 Other Resource Requirements

Parts

- Beaglebone Black
- Test stand to verify our systems are working properly
- Thermocouples
- Relays
- Project boxes
- Will need to order multiple revisions of multiple PCBs
- Time

Software Resources

- PID libraries

- Element14 community forums
- Oscilloscope analysis of original PID controller
- Nginx web server datasheets

Hardware Resources

- Analog Digital Data sheets
- NIST Thermocouple Database
- Beaglebone Datasheets
- Display Datasheet

4 Design

4.1 Design Context

4.1.1 Broader Context

Injection molding for hobbyists is a relatively new industry, but it is already quite busy. One Google search will show the number of small companies and industrious individuals who are ready to sell you their iteration on injection molding. Minufacture is unique in this space because it focuses on serving a growing niche of customers that are interested in sustainability in all parts of life. Additionally, Minufacture produces a product that is uniquely cheap, and we hope to honor that dedication in our additions.

4.1.2 User Needs

Injection molding hobbyists who are also interested in protecting the environment would benefit from access to our product, which is a fun and interesting way to combine those two interests. Additionally, communities interested in finding activities for a community center or library might find this a fun way to get students interested in both engineering and environmentalism.

Designers need a way to include unique plastic molded parts into their products, because the inclusion of these pieces brings a new level of design into their products.

Since this product is meant to be designed with communities in mind, our users need our interface to be intuitive and easy to learn. Since some of our users may be children, we need to have safety measures to protect them.

4.1.3 Prior Work/Solutions

PID controllers are available on the market, but most of them are standalone systems with proprietary software which make them hard to customize to suit your project, especially for temperature sensitive products. None of the PID controllers currently on the market that we found were able to control two separate temperature setpoints.

4.1.4 Technical Complexity

Below is the four different subsystems of our project and the skills required for each subsystem

a. Display
    i. UI programming
b. Circuit board
    i. Embedded systems
    ii. Electronics
    iii. PCB design
    iv. Soldering
c. Microcontroller,
    i. Embedded systems
    ii. Python Programming
d. Recording Data and sending it to the cloud,
    i. IOT devices.
    ii. Cloud services

We did some research on other options for hobby injection molding, and while they exist, our project will expand their capabilities. Having an onboard microcontroller that can be reprogrammed with different settings for different plastics is a new feature, and our use of the cloud to increase accuracy further once data on performance is collected is certainly a new feature. Also having the ability to control two different heating elements at different temperatures is a feature that does not appear to be on the market currently.

4.2 Design Exploration

4.2.1 Design Decisions

Design decisions including, but are not limited to, materials, subsystems, physical components, sensors/chips/devices, physical layout, like.

1. Display
2. Microcontroller
3. Pre-made/homemade Thermocouple circuit

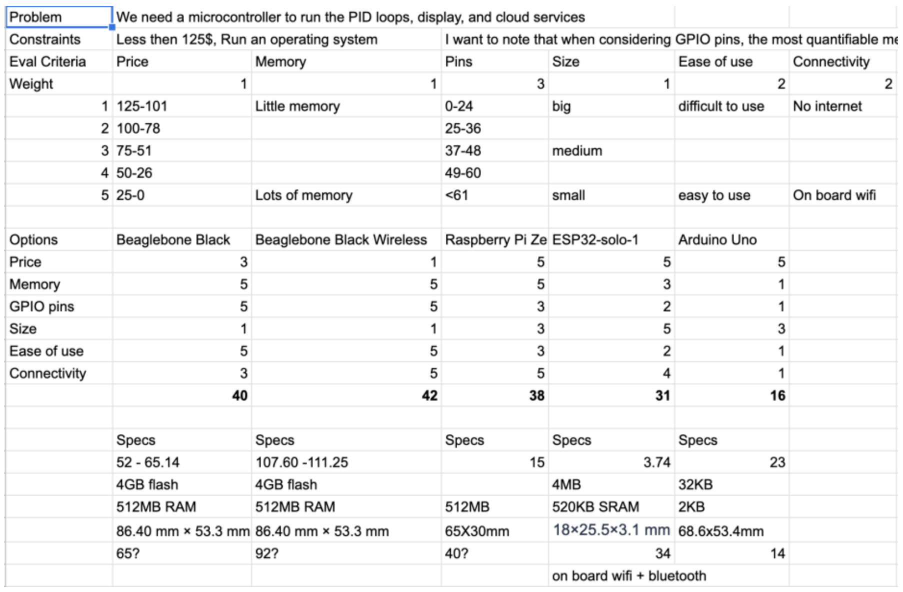
4.2.2 Ideation

For the display we searched on digikey for displays that matched our constraints.

- Mikroe-55
- Mikroe-495
- E35KA-FW1000-N
- PH480272T005-IHC03

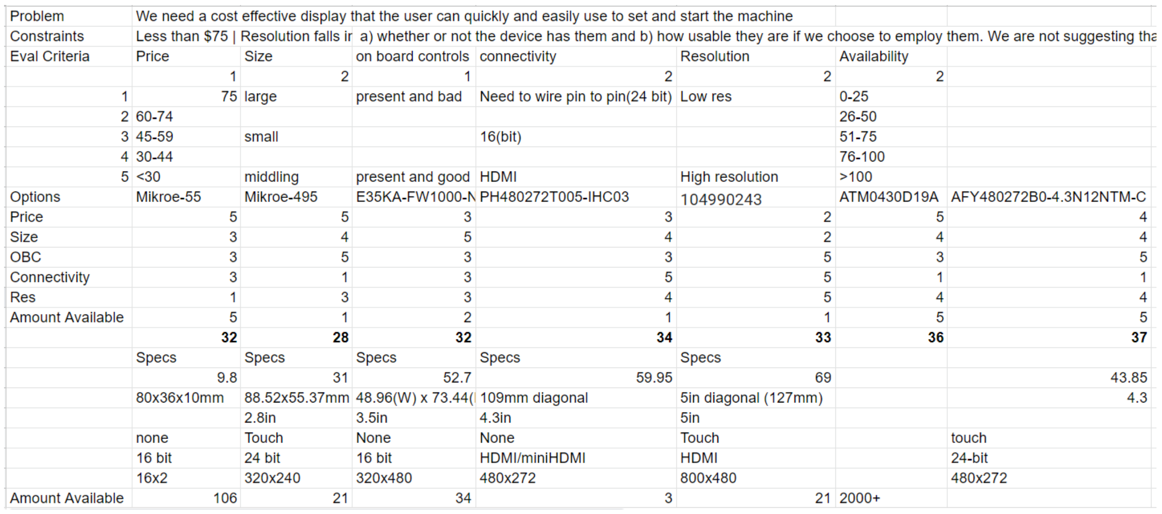
- 104990243
- ATM0430D19A
- AFY480272B0-4.3N12NTM-C

4.2.3 Decision-Making and Trade-Off



Problem	We need a microcontroller to run the PID loops, display, and cloud services					
Constraints	Less then 125$, Run an operating system		I want to note that when considering GPIO pins, the most quantifiable me			
Eval Criteria	Price	Memory	Pins	Size	Ease of use	Connectivity
Weight	1	1	3	1	2	2
1	125-101	Little memory	0-24	big	difficult to use	No internet
2	100-78		25-36			
3	75-51		37-48	medium		
4	50-26		49-60			
5	25-0	Lots of memory	<61	small	easy to use	On board wifi
Options	Beaglebone Black	Beaglebone Black Wireless	Raspberry Pi Ze	ESP32-solo-1	Arduino Uno	
Price	3	1	5	5	5	
Memory	5	5	5	3	1	
GPIO pins	5	5	3	2	1	
Size	1	1	3	5	3	
Ease of use	5	5	3	2	1	
Connectivity	3	5	5	4	1	
	40	**42**	**38**	**31**	**16**	
	Specs	Specs	Specs	Specs	Specs	
	52 - 65.14	107.60 -111.25	15	3.74	23	
	4GB flash	4GB flash		4MB	32KB	
	512MB RAM	512MB RAM	512MB	520KB SRAM	2KB	
	86.40 mm × 53.3 mm	86.40 mm × 53.3 mm	65X30mm	18×25.5×3.1 mm	68.6x53.4mm	
	65?	92?	40?	34	14	
				on board wifi + bluetooth		

Figure 4



Problem	We need a cost effective display that the user can quickly and easily use to set and start the machine						
Constraints	Less than $75 | Resolution falls ir	a) whether or not the device has them and b) how usable they are if we choose to employ them. We are not suggesting tha					
Eval Criteria	Price	Size	on board controls	connectivity	Resolution	Availability	
	1	2	1	2	2	2	
1	75	large	present and bad	Need to wire pin to pin(24 bit)	Low res	0-25	
2	60-74					26-50	
3	45-59	small		16(bit)		51-75	
4	30-44					76-100	
5	<30	middling	present and good	HDMI	High resolution	>100	
Options	Mikroe-55	Mikroe-495	E35KA-FW1000-N	PH480272T005-IHC03	104990243	ATM0430D19A	AFY480272B0-4.3N12NTM-C
Price	5	5	3	3	2	5	4
Size	3	4	5	4	2	4	4
OBC	3	5	3	3	5	3	5
Connectivity	3	1	3	5	5	1	1
Res	1	3	3	4	5	4	4
Amount Available	5	1	2	1	1	5	5
	32	**28**	**32**	**34**	**33**	**36**	**37**
	Specs	Specs	Specs	Specs	Specs		
	9.8	31	52.7	59.95	69		43.85
	80x36x10mm	88.52x55.37mm	48.96(W) x 73.44(	109mm diagonal	5in diagonal (127mm)		4.3
		2.8in	3.5in	4.3in	5in		
	none	Touch	None	None	Touch		touch
	16 bit	24 bit	16 bit	HDMI/miniHDMI	HDMI		24-bit
	16x2	320x240	320x480	480x272	800x480		480x272
Amount Available	106	21	34	3	21	2000+	

Figure 5

We did trade studies to help us choose our microcontroller and display for our system. As you can see above, the results of our trade studies helped us determine which features were important and which ones didn't have as much of an effect on our choice.

4.3 Initial Design

This section refers to the design created in 491. During implementation, a lot of things changed, which will be discussed in section 6 Implementation.

4.3.1 Design Visual and Description

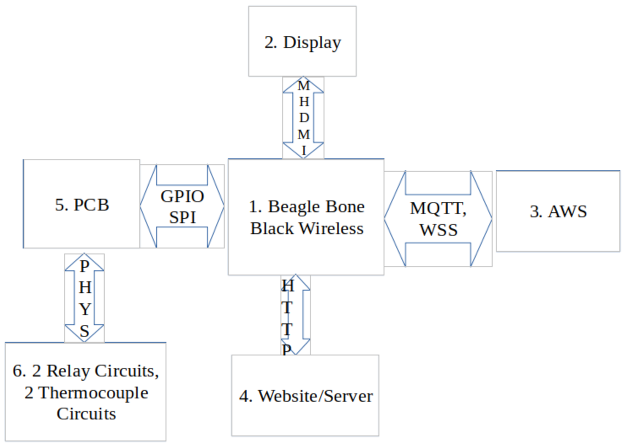

2. Display
HDMI
5. PCB
GPIO SPI
1. Beagle Bone Black Wireless
MQTT, WSS
3. AWS
PHYS
HTTP
6. 2 Relay Circuits, 2 Thermocouple Circuits
4. Website/Server

Figure 6

Our proposed design is to create a PCB cape for a Beaglebone black. The cape will have circuits to control the relays, read from the thermocouples, and provide power to the Beaglebone until shutdown is complete. The Beaglebone will have a python GUI program that will also have an underlying command line to allow for future changes of the GUI. The Beaglebone will also be able to connect with a web UI and AWS. We plan to use the AWS connection to store values like runtime, types of plastic used, and mold used. Our client will monitor these values and use them when considering the future project direction.

4.3.2 Functionality

Our initial design is meant to seamlessly combine an easy-to-understand UI and a highly responsive PID protocol which will keep the heating element within 10 degrees of its goal. Should our design achieve everything that it’s meant to, a user will be able to select a mold (provided by our client), slot it into the machine, and press ‘go.’ Plastic in a funnel above the machine will be melted and manually pushed into the mold, from which the creation can be pulled out safely after a cool down period.

4.3.3 Areas of Concern and Development

Due to unexpected issues and time constraints, we have not been able to do as much full system testing as we would have liked to do. We are concerned that there might be edge cases in the hardware or software that might trigger an unexpected failure event when in use. This can be resolved through some more extensive testing of the system by the Minufacture team as they begin to implement our work into their full product.

4.4 Technology Considerations

As we talked about in section 4.2.3 Decision-Making and Trade-Off, we conducted trade studies (Figures 4 and 5) to determine which display and microcontroller we would use. For the microcontroller we considered the price, memory, GPIO pins (as in how many there were, and what built-in options existed), size, ease of use, and connectivity (as in how easy it would be for us to integrate it into the rest of our system. For the display we considered price, size, connectivity, resolution, and availability. We scored options from the market based on what we thought was most important for each and used those scores to select our parts.

4.5 Design Analysis

Our design proposal from section 3.3 has worked out so far since we started out with research, moved into implementation. Once our PCB arrived, we went ahead with the integration and began testing. We've had to do two revisions and we are currently finalizing our testing and debugging. Overall, we've been able to have a system that can set and track the temperature, edit and save profiles, display data on GUI and do a gentle shutdown.

4.6 Design Plan

Our design plan used the milestones mentioned in section 3.3 and our meetings with our client to determine what needed to be worked on in each sprint. The interfaces on our designs are between the PCB and the display, the PCB and the Beaglebone Black and the PCB and the thermocouples and relays.

5 Testing

5.1 Unit Testing

We have three major parts of our system that need to be tested. They are:

- The GUI
  - ⊄ Tested through software and use of system
  - ⊄ Underlining command line interface makes it easier to understand the code and makes debugging easier.
- The PID Controllers
  - ⊄ Verify over multiple testing cycles that it stays within 10 degrees of setpoint including for different temperature profiles
  - ⊄ Various testing methods with Beaglebone Black
- PCB
  - ⊄ Ensure safe operation at required power levels for each board.
  - ⊄ Test individual circuits and subsystems for desired continuity and operation.
    - ■ Troubleshoot issues found.
  - ⊄ Test the entire system.

5.2 Interface Testing

In our design, there are four important interfaces. The first is from the Beaglebone to the PCB. This interface will be tested with a function that we will write into the firmware. This function will test every important solder connection and throw an error light if it fails. The second important interface is between the PCB and the display. This interface will be tested by making sure that there are no shorts between any of the pins from the connector to the PCB, and by observing the pins with an oscilloscope while the system is powered on. The third interface will be between the thermocouples and the PCB. When the thermocouple sensors are not plugged in, the voltage from the connector will be ~0V. When they are plugged in, the voltage will be ~0.7V. The PCB will throw an error light if it senses a poor connection.

5.3 Integration Testing

Our client has provided us with a test stand that simulates the integration with the current system. Once we have built a control structure that can accurately control that test stand, our client will integrate it with the actual product. As such, our project's scope will not include integration testing.

5.4 System Testing

For system level testing we are going Verify that a user can set, create or edit a temperature profile with the GUI. When ready, the system will control the heating relays and react based on the temperature read from the thermocouple.

5.5 Regression Testing

For physical PCBs, we will keep old PCB revisions in stock and keep our old PCB design revisions in our git repository. For our software work, we will continue to store old versions of the code in our git repository and will reference them should we make changes which cause failures.

5.6 Acceptance Testing

For our design requirements to be met, we must show that the thermocouple is able to detect the temperature required for the designed profiles to within less than 10 degrees of the desired temperature.

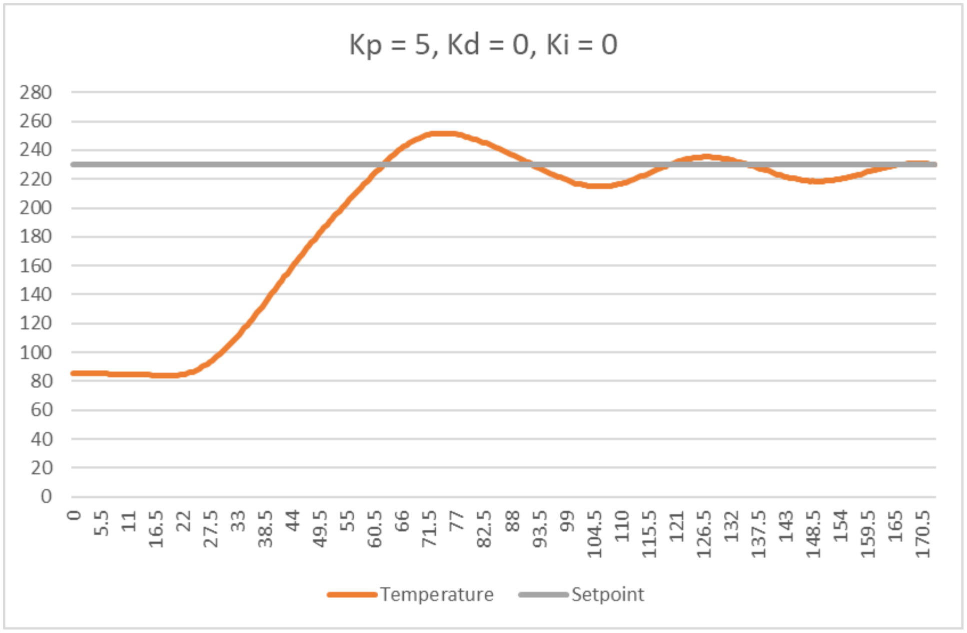
Secondly, the PID control software needs to ensure that the temperature is maintained within the needed range by holding the relay at the desired temperature of between 150 and 450 degrees. Thirdly, the Beaglebone controller will be able to display the profiles, temperature, and system status via integrated UI. We interface with our client, when possible, to ensure that he approves of our technical decisions and is pleased with progress.

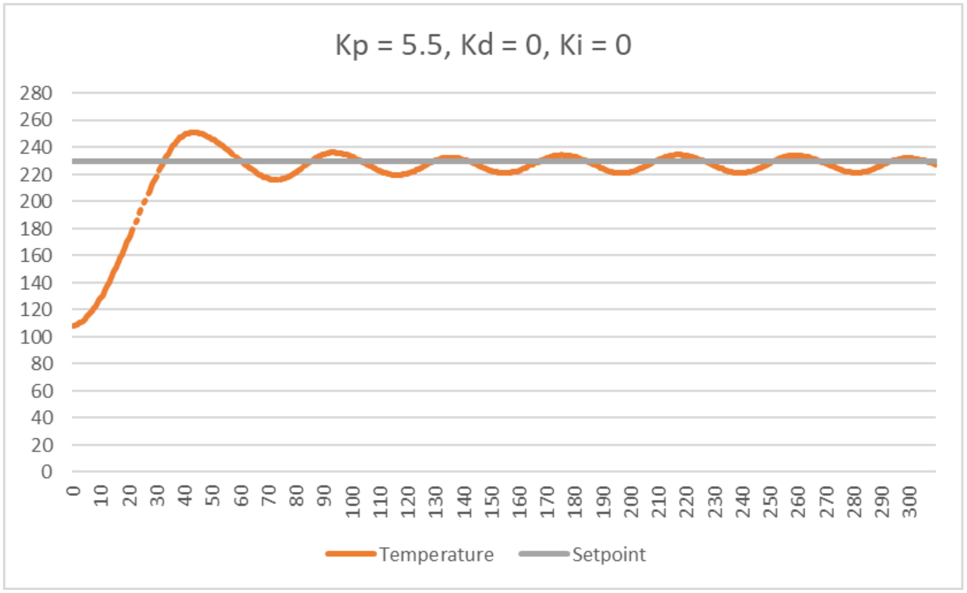
5.7 Results

Through our unit testing we were able to diagnose most of the issues we had in our first two prototypes. We have also been able to fix most of the bugs in the UI codes.

During our system testing we were able to confirm that profiles could be chosen, set points set, and the heating elements were toggled appropriately.

While we did not get to do as much acceptance testing as we had wanted to, after the initial overshoot we were able to hold the temperature right around the setpoint. After the second cycle our temperatures are within the +- 10 degree range.





6 Implementation

6.1 Hardware

6.1.1 Thermocouple Circuit

In the first iteration of the thermocouple circuit, we tried to make our own circuit. This circuit made it onto our first version of the PCB. The circuit did not work. There were a lot of different issues on the board, so we were not able to figure out exactly why it did not work.

To reduce the number of potential problems on the next version of the PCB we decided to use a premade Adafruit thermocouple circuit board. We had the intention of putting this circuit back on the board, but we were not able to get that done.

6.1.2 Relay Circuit

When we first implemented the relays, they were controlled by the Beaglebone directly and powered by 3.3V. However, as we tested this design, we found that the relays would switch but could not maintain their position. Looking deeper into the data sheet we discovered that the relays required 0.45W, this value was well above the capability of the beagle bone pins to supply.

To solve this problem, we changed the relays to be powered by 5V so they could pull from our main power supply. We also hooked up a low side control MOSFET with its gate connected to the Beaglebone Blacks pin. This allows us to supply all the power we need while still controlling the relay from the Beaglebone. However, we did find a new issue in this revision. The relays were on from bootup and only turned off when they were configured. This proved to be a safety hazard, so we fixed it in our final revision.

We were able to find a simple solution in our final revision. We realized that our previous revision had the gate of the control MOSFET floating instead of being pulled to ground. Thus, all we had to do was a pull-down resistor to the gate of the MOSFET.

6.1.3 Gentle Shutdown

One of the challenges we face when using microcontrollers like Beaglebone is that if the Beaglebone loses power unexpectedly while transferring files it can corrupt the operating system. Thus, we needed a way to be able to continue providing power to the Beaglebone as it closes all the running processes. To do this we decided to add a battery system to our power circuit.

Our original thinking to solve this problem was to add enough capacitance on the 5V line to hold the Beaglebone on while the shutdown process finished. In our initial test, we needed to hold the Beaglebone at 5V +- 0.25V for at least 10 seconds, while the Beaglebone was drawing 0.2A. After some calculations and testing we concluded that using capacitors was not workable as we would need a large number of Farads, and this large amount would slow down our boot up time drastically as the capacitors charged.

So, for our first PCB design we used a 9V battery and a relay circuit to switch from line voltage to the 9V power. While this circuit worked it was large, expensive, and not rechargeable (which was important to our client).

For our next iteration, we switched to a LIPO battery with a charging circuit and a power OR-ing IC. The Power OR-ing circuit allowed us to detect when line voltage went down and switch the input of our 5V switching regulator to the LIPO battery voltage. Once the Beaglebone was fully shut down it shut off the LIPO voltage. This change to a LIPO battery required our 5V regulator to be changed to a Buck/Boost configuration. This configuration worked very well except that we had LIPO voltage leaking though the body diode of our control MOSFET, making us unable to shut down the LIPO battery when the Beaglebone fully shut down.

To solve this, we added a second control MOSFET in series with the first in the reverse direction. This blocked the leakage and allowed our circuit to fully function.

6.2 Software

6.2.1 GUI

For the GUI we used a python library called Tkinter since it is lightweight, easy to program, and has a lot of features that we could use in our design. The GUI allows the user to create, and edit various profiles needed to melt various kinds of plastic. The GUI communicated with an underlying command line interface to do things such as accessing the databases to work with profiles, PID values, and run the main program. The GUI is easy to understand and should open on screen when the system is powered up after the setup script has been run.

6.2.2 Command Line Interface

As mentioned in the section above, the command line interface handles SQL query's, reading the thermocouples, and turning on the relays when needed. We decided to implement the system this way for two reasons. The first was that we were advised to do so from our faculty advisor. The second reason is it helps separate our GUI from the logic so that anybody can figure out what our code is doing and make changes to it as needed. This was further driven by the fact that our project will be open source after we are done with it and that there is a community of people that have similar, but not identical machines.

6.2.3 Device Tree Overlay

Since we are using a display that is not directly sponsored by the Beaglebone Black platform we had to create our own device tree overlay to configure the pins needed to control the display. The overlay also sets other things such as the resolution, timings of frames, and clock frequencies.

This part of the project is locked to this image of the Debian OS and this display. While it might work with other displays at first there may be issues the occur such as the screen flashing white, the resolution being off the GUI no longer fits on the screen.

6.2.4 System Setup Scripts

We wrote a series of scripts that would allow the user to make the system useable from the box within a short amount of time by running a single bash script. The script updates the images, sets up the databases used to store the profiles, history and PID values, sets up a script to run at boot to configure the pins needed for the SPI. The details of how to use this script are provided in a README file. That file also informs the user how to connect the Beaglebone to the internet on various operating systems. If the script is successfully run, the system will be completely ready to use after a reboot.

6.2.5 PID Autotune

We worked on an autotune program for the PID controller. While we are not able to get this implemented into our final various, we are going to send it along to our client so that they can add it if they would like. The program runs for a few cycles to a specific point. Based on the speed that it takes the system to reach the setpoint the gains of the PID controller are changed. Once all the cycles have been run the program returns the new PID values.

7 Closing Material

7.1 Conclusion

After two semesters of hard work, we are proud of what we have accomplished. The PID controlled heating element works very well, and it stays well within our 10-degree range requirement. Our Beaglebone Cape is reliable, safe, and size efficient. The software is well written, robust, and commented on in such a way that anyone in the future could understand how it works with very little effort. From working on this project, we all learned and honed many skills both technical and managerial. While we are disappointed that we were unable to deliver a complete product to our client, we are confident that the work we have done so far provides a great step forward and will be an excellent addition to Minufacture's injection moulder in time.

7.2 References

"Beaglebone Black," *Beagle Board - beagleboard.org*. [Online]. Available: https://beagleboard.org/black. [Accessed: 08-Dec-2022].

"The Best DIY & Desktop Injection Molding Machines of 2021," *All3DP*, 02-Apr-2022. [Online]. Available: https://all3dp.com/2/desktop-diy-injection-molding-machine/. [Accessed: 08-Dec-2022].

Bobby100 and mybeer2018, *Omega Engineering*, 31-Jan-2022. [Online]. Available: https://www.omega.com/en-us/control-monitoring/controllers/pid-controllers/p/CN8200-Series. [Accessed: 08-Dec-2022].

"FT3403 economic LCD Digital Intelligent PID Temperature Controller," *Ft3403 Economic Lcd Digital Intelligent Pid Temperature Controller - Buy Pid Temperature Controller,Temperature Controller,Digital Temperature Controller Product on Alibaba.com*. [Online]. Available: https://www.alibaba.com/product-detail/Temperature-Controller-FT3403-Economic-Lcd-Digital_62136434966.html. [Accessed: 08-Dec-2022].

IEEE Standards Association, 01-Dec-2022. [Online]. Available: https://standards.ieee.org/. [Accessed: 08-Dec-2022].

"Products," *Minufacture*. [Online]. Available: https://minufacture.com/collections/all. [Accessed: 08-Dec-2022].

8 Appendices

8.1 Operation Manual

Congratulations!

On purchasing your new injection molding machine! We're so glad you chose this product. In the following instructions you will learn how to start and use the embedded controller that operates your injection molder. Follow the steps carefully, stay safe, and you'll be molding in know time.

Initial Beagle bone Black (BBB) Setup:

Before starting the default username and password for the BBB are:

Username: debian

Password: temppwd

NOTE: Make sure you do not have a space after the username.

1. Download and etch Debian image 9.5-lxqt-amhf-2018-10-7 to a SD card with at least 8 GB of storage:
    a. The image can be found at BeagleBoard.org - latest-images or in the tar folder.
    b. The balenaEtcher program can be downloaded from balenaEtcher - Flash OS images to SD cards & USB drives

2. Connect the BBB to a computer via USB and ssh into the BBB:
    a. In Linux terminal: ssh debian@192.168.7.2
    b. With PuTTY the setup should look like the image below. Make sure to hit accept the first time you connect to the BBB. In the window that appears it will ask for the username and password.
3. Connect the BBB to the internet:
    a. Steps for Linux OS over USB:
    b. Steps for Windows OS over USB: How to Connect a Beagle Bone Black to the Internet using USB (digikey.com)
    c. Steps for BBB Wireless: How to Configure the Wifi on a Beagle Bone Black Wireless – Factory Information Systems Center (gatech.edu)
4. Use scp to put the tar file onto the BBB:
    a. Please make sure to put the folder at the following file path. /home/debian/Desktop
    b. Via Linux Terminal: scp local_dir/project.tar.gz root@192.168.7.2:/home/debian/Desktop
    c. Via WinSCP: Inspire | BeagleBone Black Tutorials, Resources and Workshops (inspire-logicsupply.blogspot.com)
5. Unpacking the project:
    a. Type “cd /home/debian/Desktop” on your terminal with the ssh connection.
    b. Type “tar -xzf project.tar.gz” in the terminal.
6. Run the system setup script.
    a. Type “sudo bash setup.sh” in the terminal.
7. Connect the PCBs and display.
8. Run the GUI program by opening a qt terminal on the display.
    a. Type “python3 GUI.py” in the qt terminal.

System setup:

1. Make sure the connections between all boards, USB devices, and the BBB are secure and pushed in as far as possible.
2. Plug the BBB into the wall. The Blue LEDs should turn on and become solid before starting to flash. After a few seconds, an image of a penguin should appear on the display.
3. The powerup sequence takes a few minutes, but the GUI should appear.
4. Once the GUI appears the system is ready to use. You can now add and edit any profiles and run the system.

Running the system

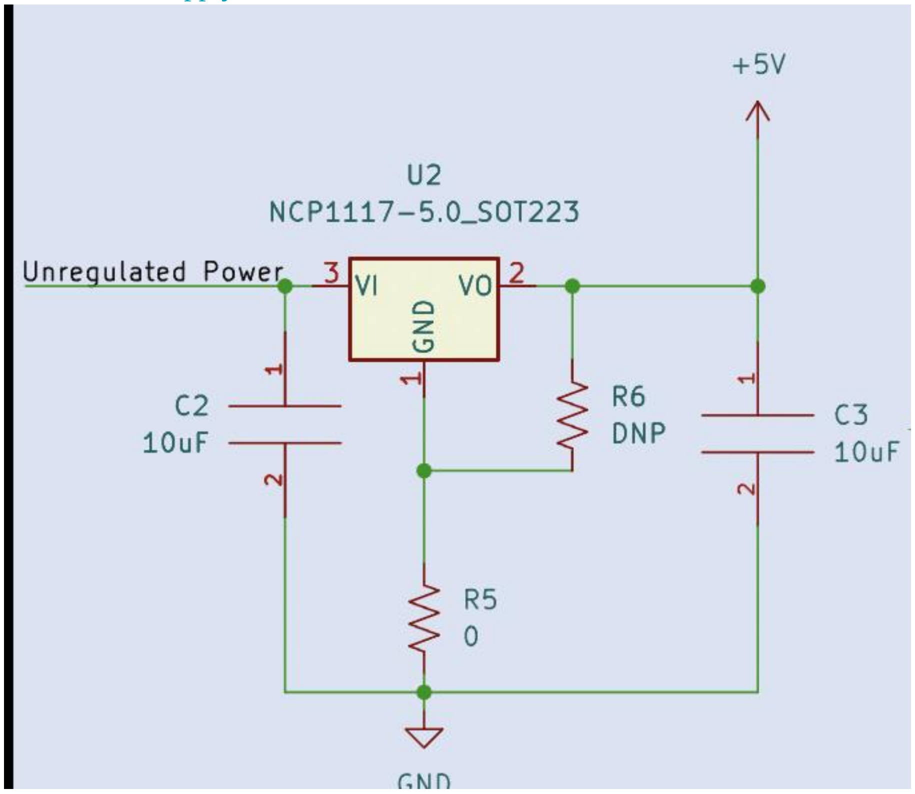
1. Run a profile
    a. From the Menu screen click Start
    b. Choose the type of plastic from the profiles dropdown
    c. Click start
        i. This will bring you to the run screen and start the PID loop
2. Editing Profiles
    a. From Menu Screen click profiles
    b. Select the profile to edit from the profiles dropdown

        i. Note – The original profiles are setup based on the melting temperatures of the specific type of plastic. It is not generally recommended to change this profile.
    c. Click edit profile
    d. Fill out the boxes of the new screen

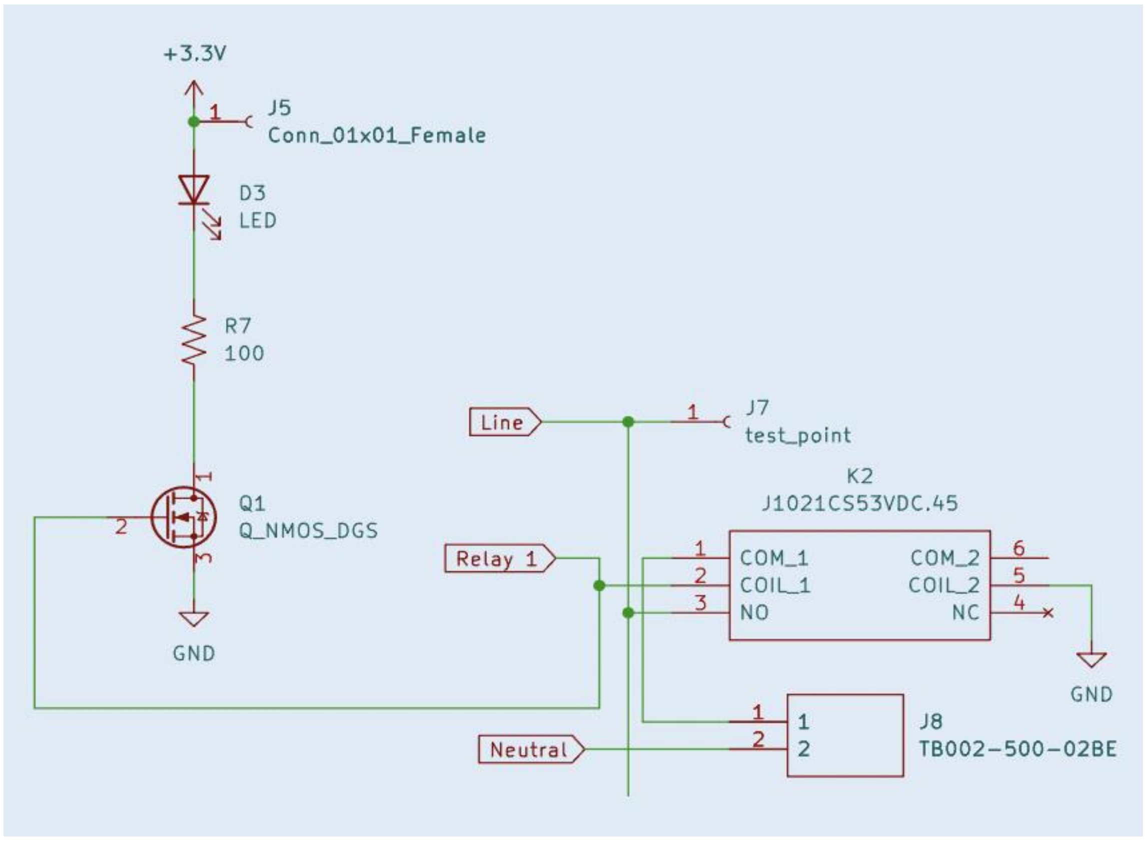
3. Creating a new profile
    a. From the Menu screen click profiles
    b. Click the new profile button
    c. Fill out each box of the new screen
        i. The new profiles track the melt time and mold type
            1. These two items are necessary to enter for tracking purposes but are not currently used in the operation of the machine
        ii. The upper and lower temperatures create the range of temperatures that the plastic will melt at
    d. Click save
4. Tuning the System
    a. From Menu screen click tuning
    b. Enter P, I, and D values
    c. Click set
    d. Run system according to section 1 and check for acceptable values

8.2 Alternative Designs

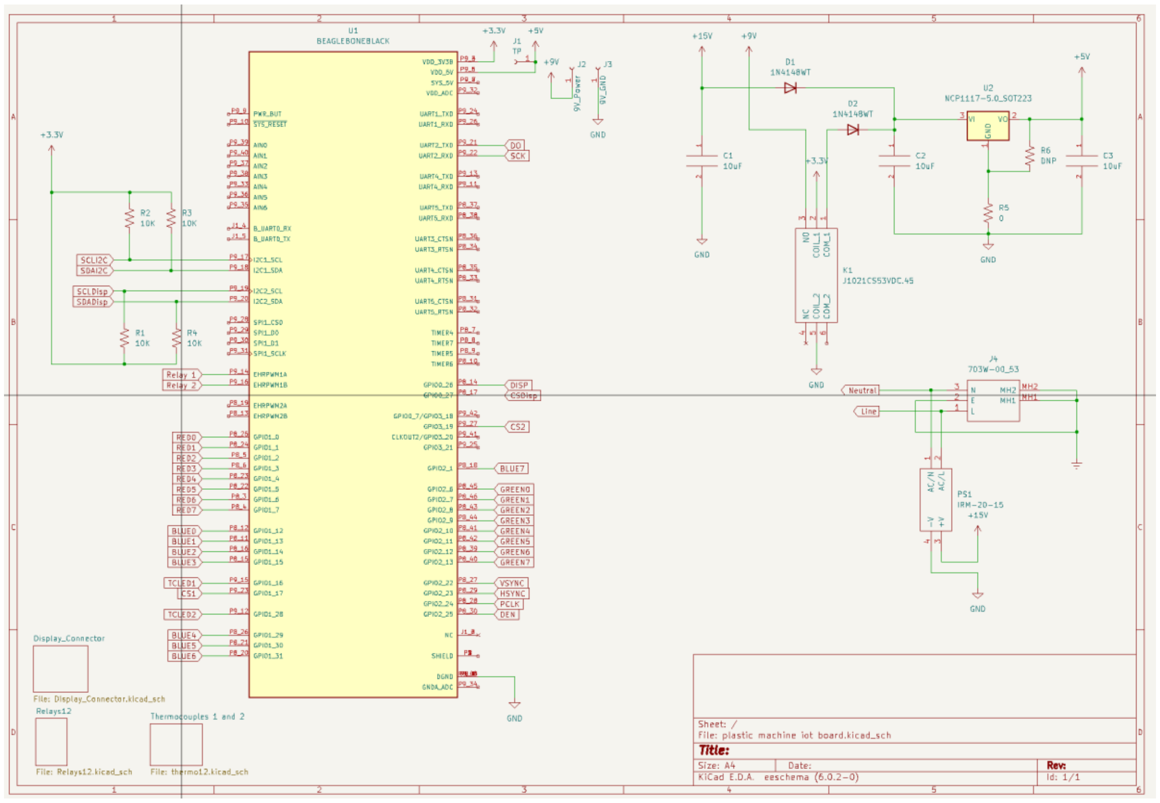
8.2.1 Power Supply Revision

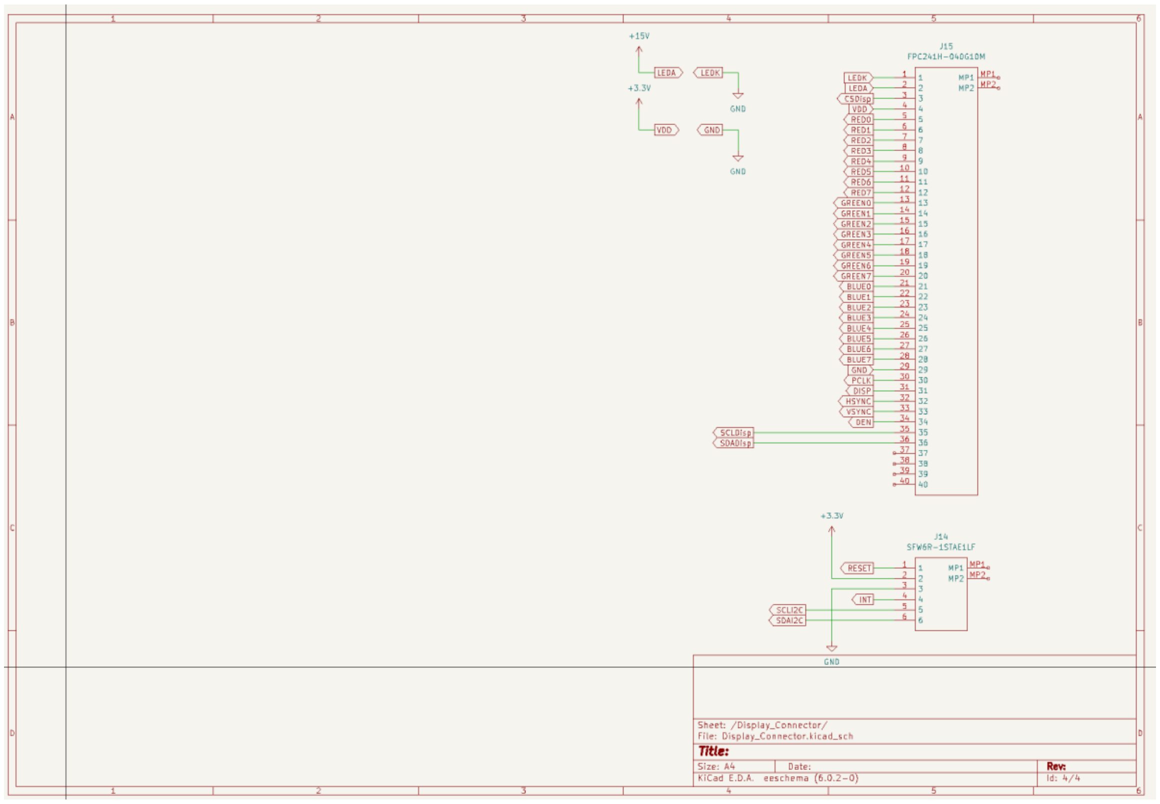


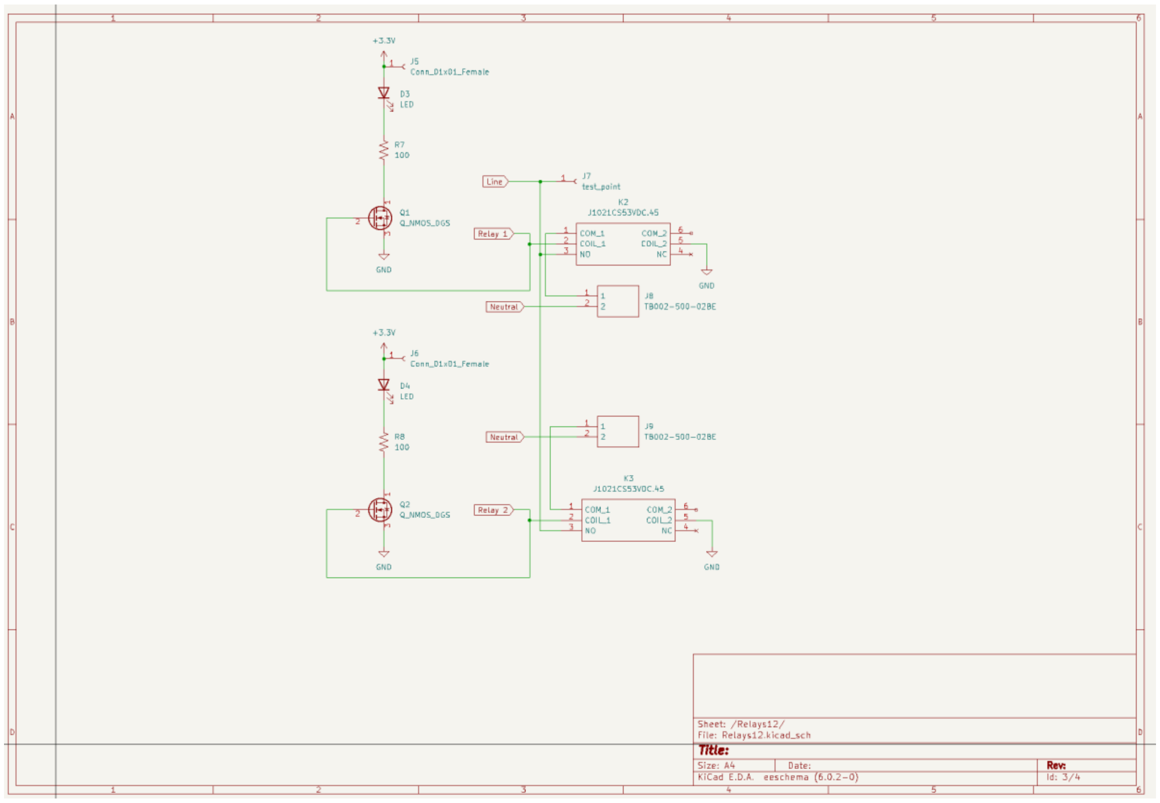
8.2.2 Relays Revision


+3.3V
1
J5
Conn_01x01_Female
D3
LED
R7
100
1
Q1
Q_NMOS_DGS
2
3
GND
Line
1
J7
test_point
K2
J1021CS53VDC.45
Relay 1
1
COM_1
2
COIL_1
3
NO
COM_2
6
COIL_2
5
NC
4
GND
1
1
2
2
J8
TB002-500-02BE
Neutral

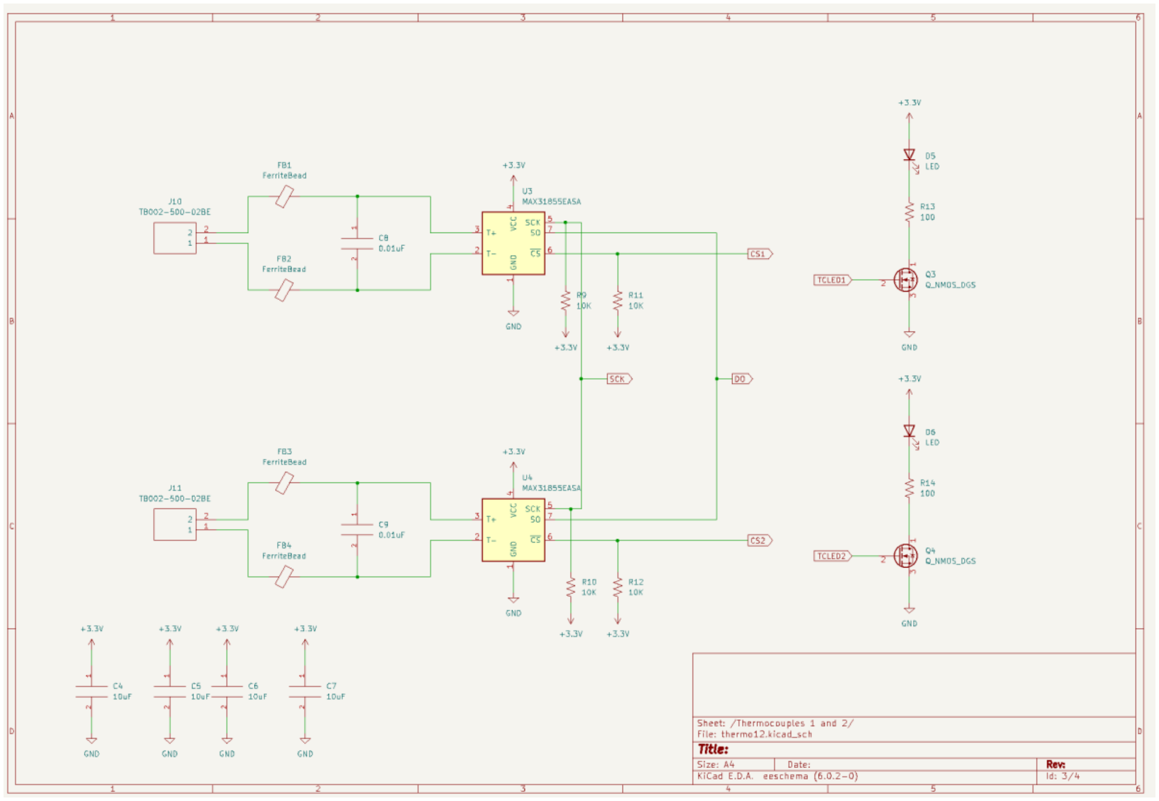
8.2.3 PCB Revision 1





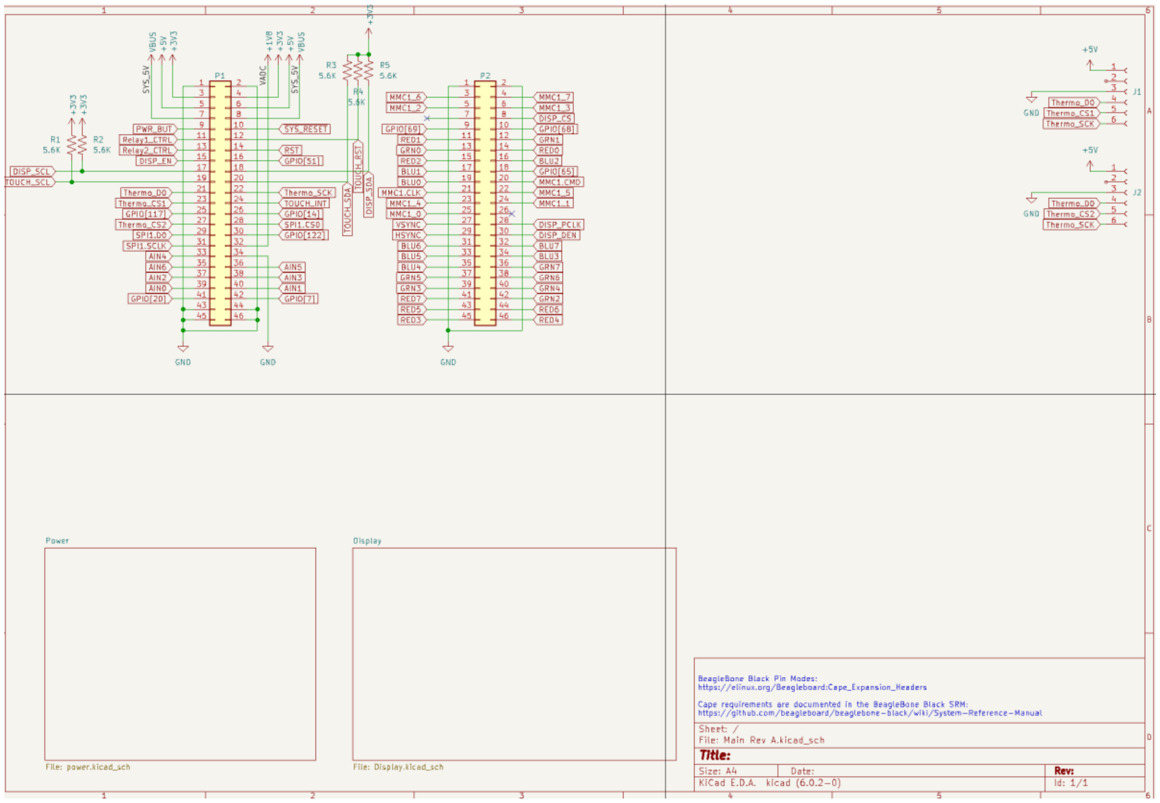


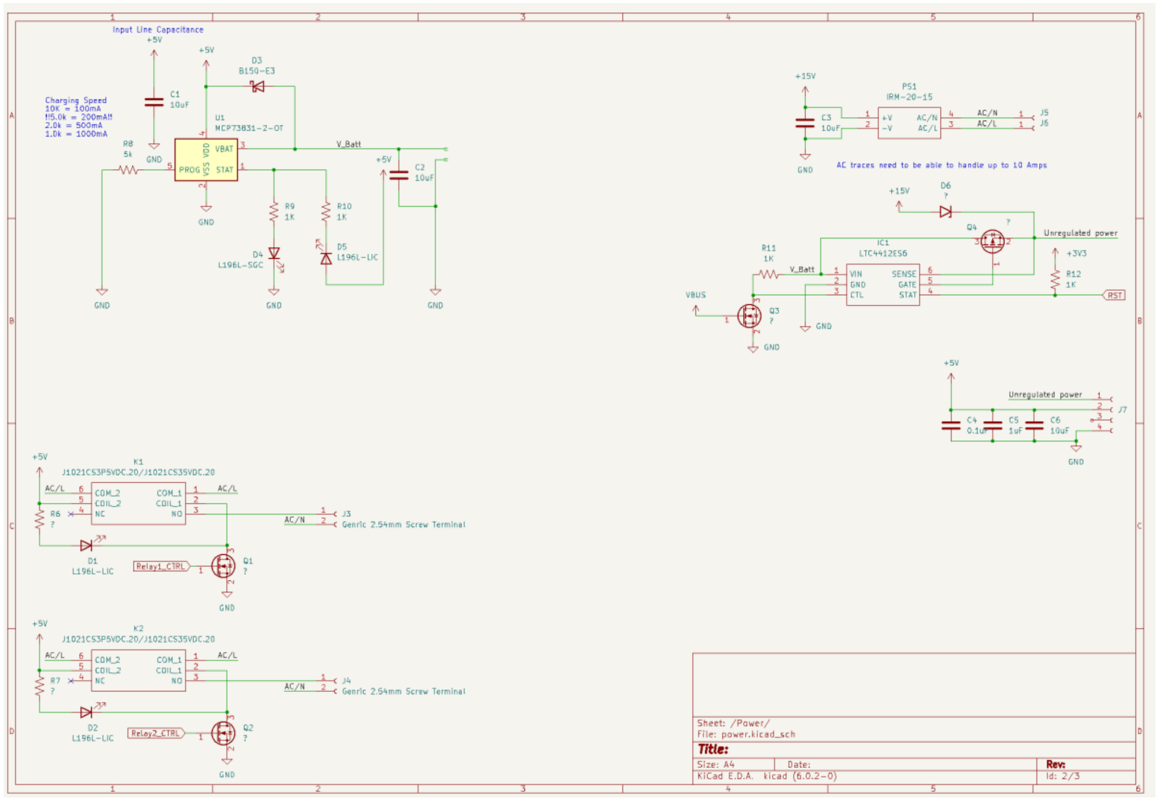
+3.3V
J5 Conn_01x01_Female
D3 LED
R7 100
Q1 Q_NMOS_DGS
GND
Line
J7 test_point
K2 J1021CS53VDC.45
Relay 1
COM_1 COIL_1 NO
COM_2 COIL_2 NC
Neutral
J8 TB002-500-02BE
+3.3V
J6 Conn_01x01_Female
D4 LED
R8 100
Q2 Q_NMOS_DGS
J9 TB002-500-02BE
K3 J1021CS53VDC.45
Relay 2
GND
Sheet: /Relays12/
File: Relays12.kicad_sch
Title:
Size: A4 Date: Rev:
KiCad E.D.A. eeschema (6.0.2-0) Id: 3/4

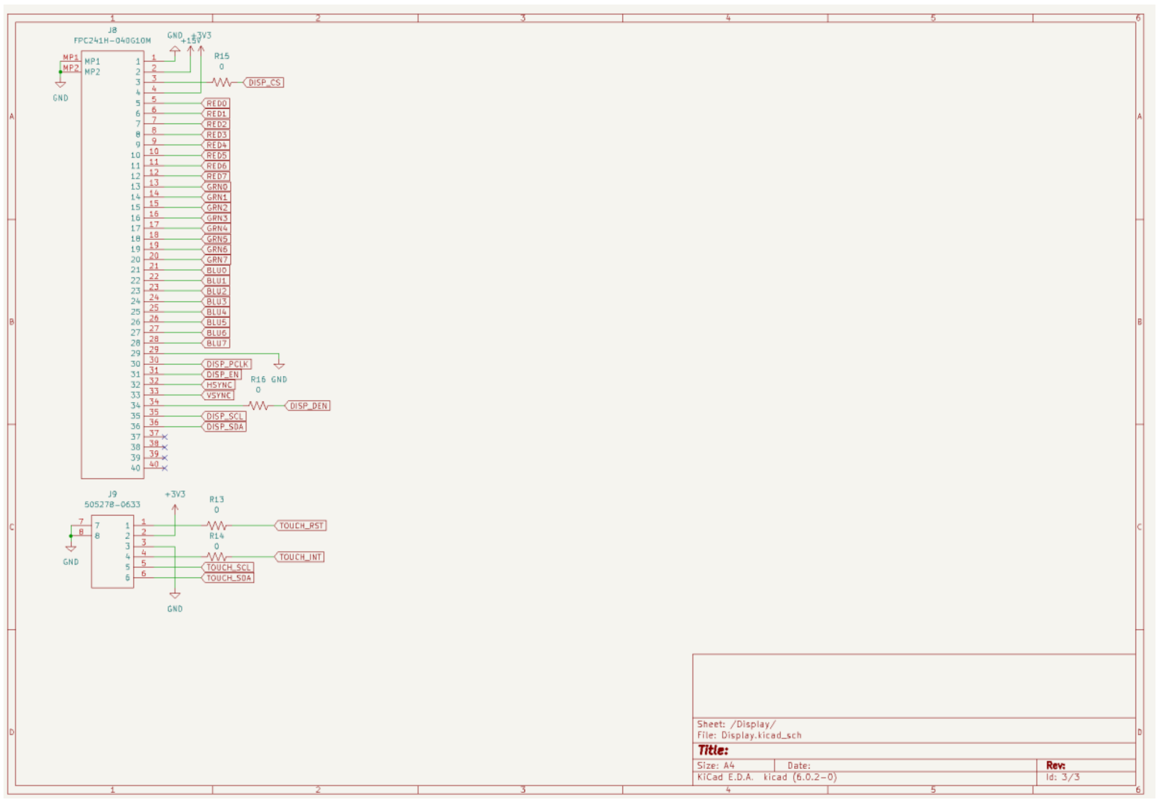


J10 TB002-500-02BE
FB1 FerriteBead
FB2 FerriteBead
C8 0.01uF
+3.3V
U3 MAX31855EASA
T+ T- VCC GND SCK SO CS
R9 10K
R11 10K
CS1
SCK
DO
D5 LED
R13 100
TCLED1
Q3 Q_NMOS_DGS
J11 TB002-500-02BE
FB3 FerriteBead
FB4 FerriteBead
C9 0.01uF
U4 MAX31855EASA
R10 10K
R12 10K
CS2
D6 LED
R14 100
TCLED2
Q4 Q_NMOS_DGS
C4 10uF
C5 10uF
C6 10uF
C7 10uF
GND
Sheet: /Thermocouples 1 and 2/
File: thermo12.kicad_sch
Title:
Size: A4 Date: Rev:
KiCad E.D.A. eeschema (6.0.2-0) Id: 3/4

8.2.4 PCB Revision 2







8.3 Lessons Learned

- Parts Research
    - We learned the importance of thoroughly researching the potential issues that a specific part can cause before choosing it. Through our trade study we focused on technical specs. We did not extensively research the problems that people face with Beaglebones and had we done so, we would not have choosen this microcontroller.
- Parts Procurement
    - We learned the importance of make sure we have backup parts especially during testing phase because, parts get damaged, blown or lost during assembly and debugging stages. While we were able to get backups eventually, we lost quite a bit of time from not having backups on hand.
- PCB Design
    - We learned the importance of triple checking the footprint size and rotation. When making the PCB there are tons of different parts in all different sizes. It was very easy to choose the wrong one or rotate it wrong.
    - We learned that the switching buck/boost circuit is highly layout dependent.
- Software Design

- It takes a lot longer to integrate all the modules together because there are probably problems in the system that will not manifest until you are trying to put everything together. For example, the device tree overlay was configured to use one of the chip select pins for the SPI bus. This wasn't noticed until we needed to put everything together since we tested the display configuration and SPI reading at different points in the semester.