Low-Cost Solution of Thermal Cycler for LifeFoundry, Inc.

By
Shaoyu Meng
Pei Liu
Yuanjiu Hu

Final Report for ECE 445, Senior Design, Fall 2017
TA: Xinrui Zhu

Dec. 4th, 2017
Project No. 1

Abstract

The report recapitulates the design, prototype, development, and integration of hardware/software to build a polymerase chain reaction (PCR) machine. This machine is mostly used for DNA segments amplification in the field of bioengineering research and development. The finished product is programmable and capable of heating DNA samples up to 92°C and cooling them down to room temperature for 30 continuous cycles. We also implemented wireless communication with mobile application using Bluetooth Low Energy (BLE) as an extra feature to provide remote feedback to the end user.

Table of Contents

1 Introduction

1.1 Purpose

Our team proposed to design and build a prototype thermal cycler for polymerase chain reaction (PCR) for a campus startup company, LifeFoundry Inc. The motivation behind this project is that current commercial thermal cyclers cost anywhere between $2,000 and $10,000, which become a barrier of entry for smaller companies due to their insufficient funding for such capital expenditures. For our sponsors, this issue is exacerbated by the fact that they require a large number of such machines for their future projects.

PCR is a bioengineering method in which DNA samples are copied through repeated heating and cooling cycles, and it typically requires 25 to 30 such cycles [1]. Although the idea is fairly simple, the implementation is challenging because the device needs to have precise control over the temperature. Because of applications associated with thermal cyclers, it is very important to limit errors to the tolerance range of allowed by PCR experiments. In the worst case, a surge of temperature over the designated value could destroy the DNA sample being duplicated.

1.2 High Level Considerations

First of all, the machine must have the same capacity as commercial-grade machines to properly conduct various experiments. As the industry standard indicates, a commercial thermal cycler should perform PCR on 96 samples on a standard 96-well plate; our machine must also be able to perform PCR on such a plate sitting on top of the aluminum housing.

The PCR application has a relatively high requirement for sustained and precise temperature. In addition, if the temperature errors get too large, it is likely to damage the samples. Based on technical feasibility and our sponsor's requirements, we proposed that the maximum temperature error tolerance is ±5 °C.

Cost also plays a major role in our design process. We could have just used the best material for each part and get a very decent result without much optimization. For example, if we could use copper for parts responsible for heat conduction and use custom-made thermal-isolating foam for the enclosure, it would be much easier to control the temperature. However, doing so will cost much more and defy our original purpose of

lowering the cost of such devices. Our sponsor set a development budget limit of $400 to ensure the cost-effectiveness of the product.

1.3 Block Level Description

Our PCR machine is comprised of five modules as shown in Figure 1. The power module takes 110 V AC voltage and converts it to 12 V DC voltage. 12 V is used by Peltier modules and the water pump. The voltage regulator built-in in the H-bridge board converts 12 V to 5 V and supplies power to the microcontroller unit, which in turn supplies power for its accessories such as the LCD display and thermocouple. The thermocouple module collects temperature data which is then converted into digital format through the analog to digital convert and sent to the microcontroller unit. The microcontroller unit integrates the temperature data into the PID algorithm which will calculate the required instantaneous power output for the temperature control module. The microcontroller unit adjusts the power output by sending 5 V PWM signal with different duty cycles. The temperature control unit will then heat up and cool down the system accordingly to reach the target temperature as quickly as possible with temperature overshoot controlled within 5 °C. The user interface module displays experiment data and allows users to control the system via buttons and LCD display, or Bluetooth and mobile app.

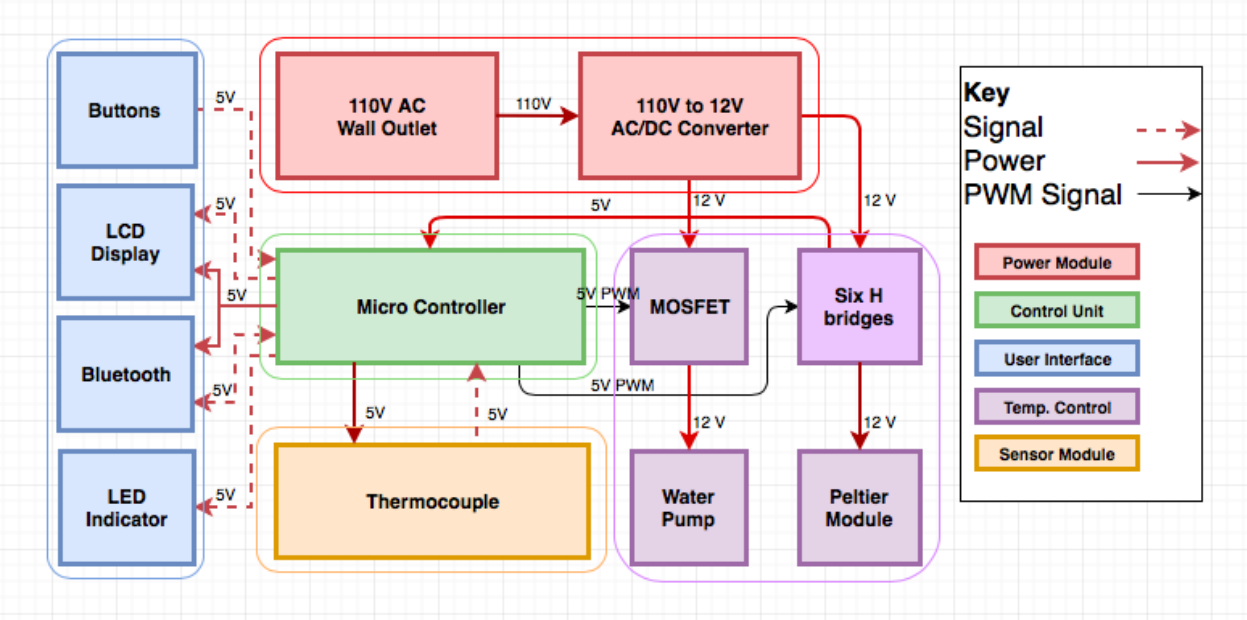


Figure 1 High Level Block Diagram

2 Design

2.1 Microcontroller Module

The microcontroller polls data from the sensor module to determine a proper duty cycle value. This is mainly done by the temperature control software via PID algorithm. The microcontroller then uses this data to control the Peltier module and the water pump, both of which work together to control the temperature. Based on the requirements of this project, the following protocols and features must be supported by the microcontroller: PWM, I2C, SPI, UART, and button inputs. ATmega328 satisfies all requirements, and it also offers a decently high clock speed, which is necessary for controlling the temperature in a real-time manner. The entire system built around the microcontroller is integrated in the printed circuit board (PCB) in Figure 12 in Appendix B.

2.2 Thermocouple Module

2.2.1 K-Type Thermocouple

We used k-type thermocouples as our temperature sensor. A thermocouple consists of two electrical conductors made from different metals. Two conductors together create a junction, which will result in voltage difference when experiencing a change in temperature. Thermocouples will be placed below the microplate, adjacent to samples, to ensure accurate temperature measurement. They then feed real-time analog temperature data to the MAX6675 chip for further processing.

2.2.2 MAX6675 SPI Interface

This chip converts the analog data directly from the thermocouples to binary data that the microcontroller can actually use. The reading has two digit precision and is send back to microcontroller using SPI protocol.

2.2.3 Calibration

The measured temperature may be different from the actual temperature due to error from the hardware and the heat loss during the heat transferring process from Peltier module to the microplate. Therefore, we measured the temperature of the sample plate using an IR thermometer, and we calibrated our thermocouple reading in reference to the IR thermometer reading. We got a result of 5 °C offset that can be fit into a linear curve.

2.3 Peltier Module

A single piece will create a transient current of 4.48 A and a steady current of 2 A. With six Peltier module together it will get 6 × 4.48 A = 26.88 A current. This current is well above the limit of a single H-bridge.

Consequently, the H-bridge design requires a change. The initial choice of H-bridge was Texas Instruments LMD18201T because it is able to provide 3 A steady current. However, the initial current is still a problem, as it triggers the thermal protection mechanism in the H-bridge. This is partially solved by connecting a 1 mH inductor in series. However, a new problem was introduced that the circuit then produced a high-pitched, audible noise caused by internal resistance of the H-bridge and the circuit.

In the end, our solution was to parallelize H-bridges [2] to power the circuit. This reduces the internal resistance of H-bridge and creates a large enough current. We also switched from using standalone H-bridge chips to using the L298N H-bridge boards. Each board contains two H-bridges each capable of driving 2.5 A of current, and eventually we ended up with 12 H-bridges (in 6 × 2 configurations) on six L298N boards.

2.4 Water Cooling module

The main purpose of the water pump is to draw excessive heat from the system by pushing water through the water block at an acceptable rate. In addition, it is necessary to add the ability to turn on or off the water pump, so that it does not run when not needed. The pump we chose was a generic fountain pump rated at 280 LPH, which is more than enough of what we need given the relatively small thermal profile of the product. The MOSFET IRF520 [3] is responsible for turning the water pump on and off.

2.5 User Interface

The purpose of the user interface module is to display the experiment data and allow user to control the device. For displaying content, the user can choose either or both of the LCD or/and the Bluetooth terminal. For controlling operations, we use three buttons to navigate through different functionalities of our device. Bluetooth control is also integrated to allow user the ability to customize PCR procedures. Figure 2 shows a flowchart of our entire user interface implementation.

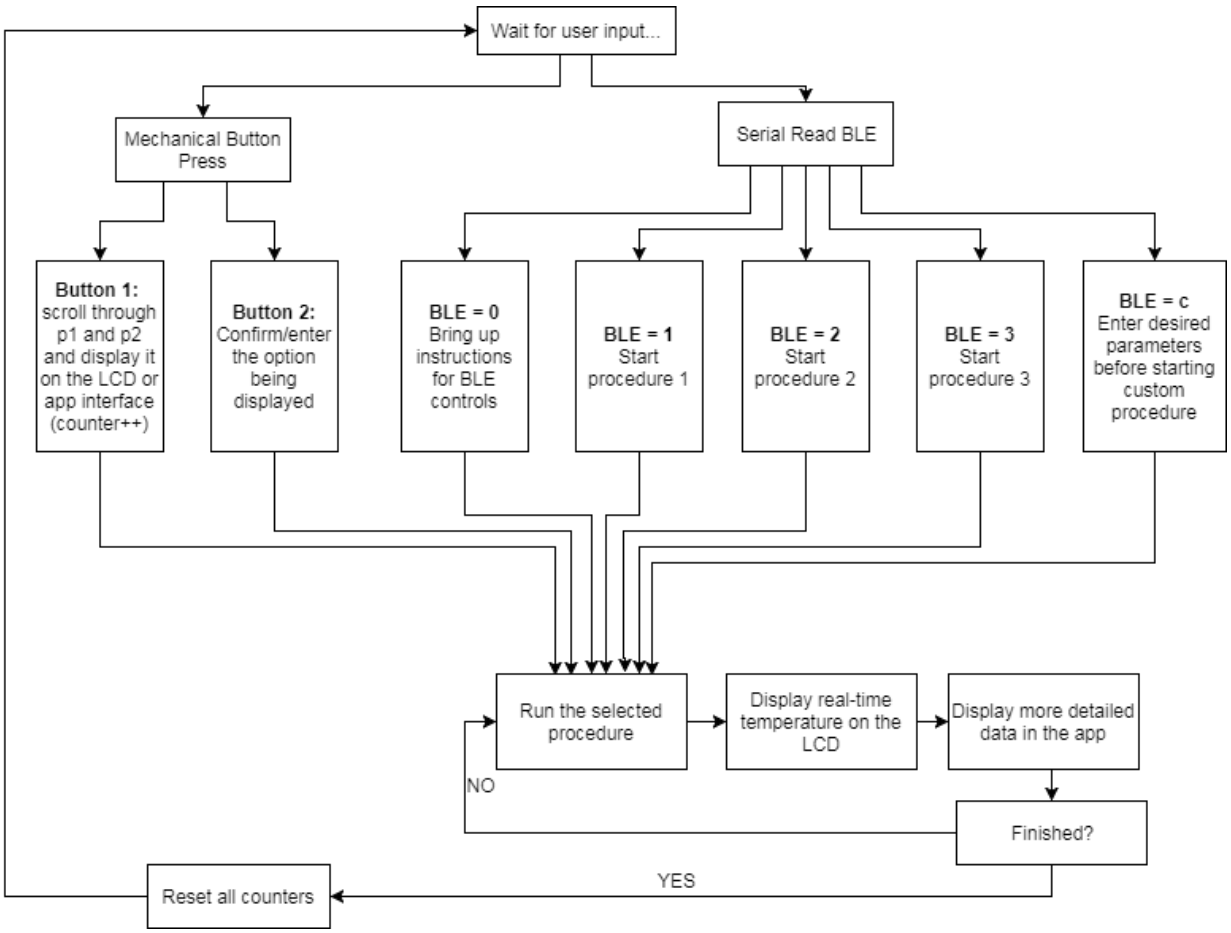


Figure 2 User Interface Flowchart

2.5.1 1602 Serial LCD Display and I2C Breakout Board

The LCD screen could display two lines of 26 characters at one time. It stores the string that needs to be displayed into its own memory buffer, and reads from its memory buffer character by character. It uses a position counter to point to the current character to be displayed and a cursor to keep track of the character's location on the screen. After the content is displayed, the memory buffer is flushed. Figure 3 shows a flowchart of the LCD software implementation.

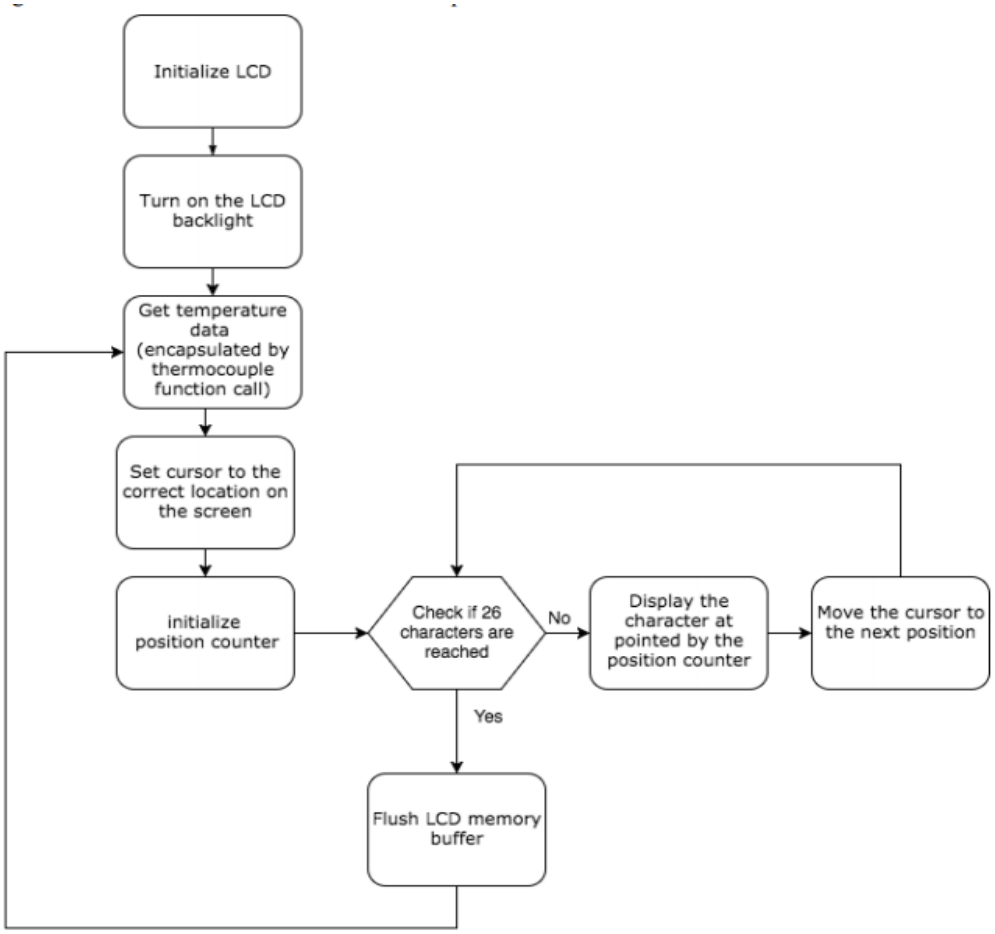


Figure 3 LCD Procedural Operations

We originally connected our LCD to the microcontroller through parallel connection. Later, an I2C breakout board was introduced to the module. The I2C breakout board used only two lines, the serial data line and serial clock line, requiring much less number of IOs from the microcontroller comparing to 8 bus lines required in the original connection.

2.5.2 Buttons

Two buttons are connected to analog input pins to the microcontroller and the third one is connected to the reset pin. Buttons were integrated in the PCB shown in Figure 12 in Appendix B.

Loop button is used to loop the menu options displayed on the LCD. The user can observe different options by continuously pressing loop button. If the user is already at the last menu option, pressing the loop button will display the first option again. The confirm

button is used to confirm the option being displayed on the LCD. Upon pressing confirm button, the device will start performing the selected procedure.

This is implemented by using two control loops in software, the first one is for displaying content on the LCD, the second one is for actually calling the procedure in the case of confirm button being pressed. The number of times loop button is pressed is stored in the buttonPushCounter variable in Figure 14 in Appendix D, and we enter procedure by using the mod operator on the counter variable.

Because buttons on the PCB are not debounced, we need a software debouncer. It is simply a small-scale finite state machine for loop button, and it detects state change when the button is pressed to prevent over-counting. It is worth noting that confirm button and the reset do not require debouncing, because they are designed to be pressed only once during each procedure.

2.5.3 Bluetooth

Because the buttons – LCD user interface provides very little flexibility for the user, we implemented a secondary user interface using Bluetooth Low Energy (BLE) as a bonus feature. With the help of Bluetooth, it is now possible to remotely operate the device. This is especially helpful if the experimenter wants to start a procedure while being outside of the lab. The experimenter will also enjoy the convenience of a bigger screen on his/her phone, along with the extra information we can display on it. Furthermore, the addition of BLE gives the experimenter the ability to fully customize the procedure on the mobile app. The user also has the option of bringing up the "help" interface by typing "0" in the app for the full instructions on how to operate the device from the app. The communication log through Bluetooth is shown in Figure 4.

The customization part is achieved using the built-in TX and RX serial pins of ATmega328 MCU. Once the "customize procedure" option is selected from the menu, the user will be prompted to enter different parameters related to the procedure. After values have been entered, the procedure will start automatically.

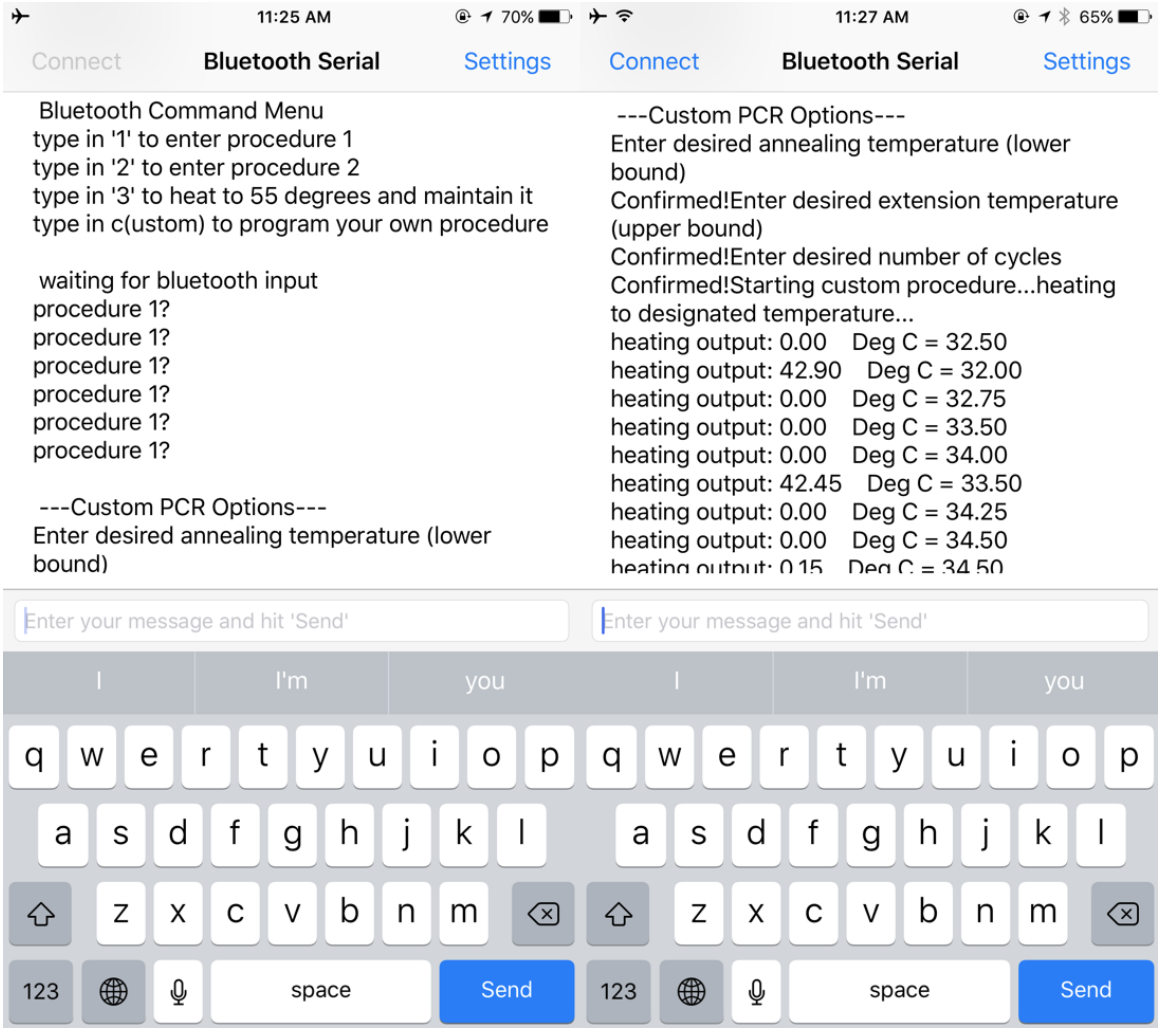


Figure 4 Mobile Application UI

2.6 Power

We chose 110 V to 12 V AC/DC converter for the power supply. The total power usage in our system at the peak output is computed below in Table 1.

Table 1 Power Consumption Calculation

Parts	Quantity	Voltage (V)	Current (A)	Power(W)
Atmega328 Microcontroller	1	5	0.019	0.095
MAX6675 SPI Interface	1	5	0.094	0.47
TEC1-12705 Peltier Cooler	6	12	4.48	322.56
Brushless Pump DC	1	12	0.35	4.2
1602 Serial LCD Display	1	12	0.35	4.2
			Total	**327.45**

2.7 Mechanical Design

We designed a variety of mechanical components including the sample plate, the plate holder, the water tank and the casing. First, we produced an aluminum plate capable of holding the standard 96-well sample plate. This is shown in Figure 5. To achieve the best thermal performance, this plate needs to have 96 holes corresponding to the 96-well plate. In addition, this part needs to be as thin and light as possible to ensure optimal thermal conductivity. This part was designed by our group and manufactured by the ECE machine shop. Second, we built a variety of 3D printed parts for different purposes, namely, a holder for holding the aforementioned aluminum plate, Peltier modules, and water block together, a power supply enclosure, and a water tank. Third, we used cardboard to construct the casing. Openings were made on the sides of the casing to ensure ventilation for the power supply, as well as accessing the buttons.

Figure 5 Aluminum Sample Block

2.8 Software Design

The software design mainly consists of two parts: temperature control and user interface. The temperature control takes the current temperature reading as an input and computes an appropriate duty cycle as an output using the proportional-integral-derivative methodology. The user interface contains a LCD, three buttons (including the reset button), and a Bluetooth module. Instructions and feedbacks are given to the user based on the status of operation, and the user can choose to perform desired procedures via buttons or the mobile Bluetooth application.

2.8.1 PID Control Algorithm

Temperature control is the backbone of our project, as it is absolutely necessary for thermal cycling. It utilizes the PID library to achieve precise control with minimum

temperature overshoot when heating/cooling to a desired temperature, as well as minimum temperature fluctuation when maintaining at set temperatures.

A PID controller is also called a proportional-integral-derivative controller [4], which is common in closed-loop control systems. A PID controller takes in a process variable (PV) and a setpoint, which is the desired value for the PV, along with three tuning parameters (Kp, Ki, Kd), and it yields an error value based on the proportional, integral, and derivative terms multiplied by their coefficients (Kp, Ki, Kd ,respectively). A basic PID controller constantly calculates u(t), the computed power output, as shown in Eqn. 1.

$$u(t) = Kp\, e(t) + Ki \int e(\tau)\, d\tau + Kd\, dt \qquad \text{(Eqn. 1)}$$

The PID Library by Brett Beauregard [5] was used in our temperature control software, which follows a very similar approach as the standard PID controller, except it returns an integer result between 0 and 255, which corresponds to all possible PWM values. In addition, it enables the programmer to choose the controller's direction. It is critical for our application of cycling because we need to use the PID controller for both heating and cooling operations. In this case, we instantiated two separate PID controllers for direct and reverse operations to match "heat", "cool", and "maintain" states. When we are trying to raise or maintain a temperature that is higher than the room temperature, the direct controller is used (virtually no PCR procedure require maintaining a temperature that is below the room temperature); when we are trying to lower the temperature (e.g. 74°C to 55°C), the reverse controller is used. Compute() runs once per PID loop to yield fresh output.

To physically switch between heating and cooling states, two inputs to the H-bridge are connected to the two PWM outputs from direct and reverse PID controllers. As shown in Figure 6, the function heatTo() is basically a wrapper for heat(), with the addition of some feedback code displaying the current PWM output level and thermocouple reading. In the beginning of heatTo(), the reading for thermocouple was reduced by 5 °C. This offset is from an experiment when we tested thermocouple readings from room temperature to around 40 °C against an IR thermometer, and found out that there is approximately a 5 °C flat offset for that temperature range.

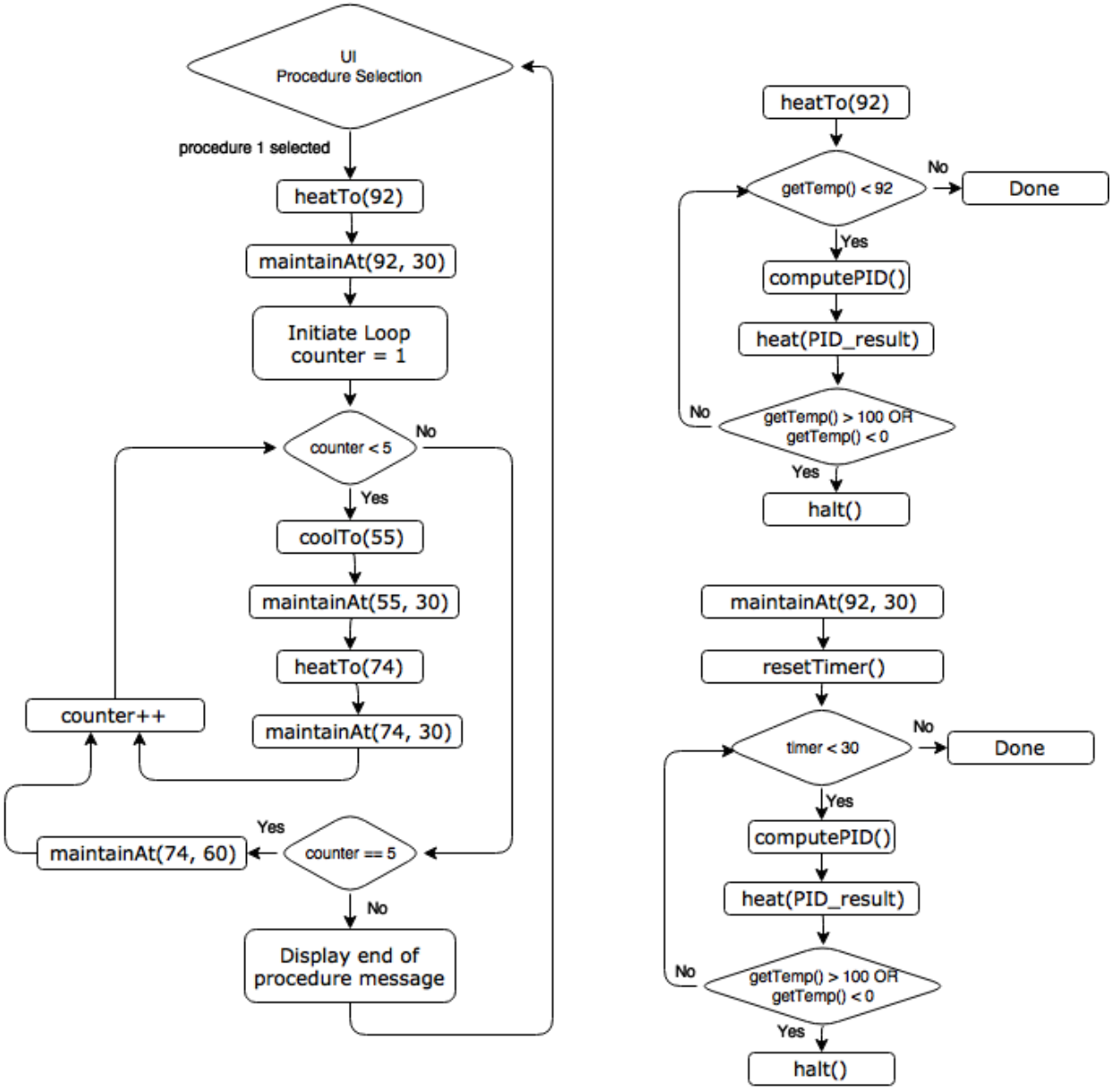


Figure 6 Temperature Control Software Flowchart (3-in-1)

Maintaining the temperature at a certain point, however, is much more difficult. Whenever there has been a change from the direct to reverse PID controller (or vice versa), the output values would temporarily get very large. This is because only one PID controller can output value and change the operation mode of the H-bridge, while both PID controllers are running at all times. When the output of reverse PID controller is being used, the direct PID controller will try to crank up the output value because temperature keeps dropping with its current "output". As soon as the output of the direct PID controller starts being used again, the output values will readjust and rapidly decrease because the thermocouple will

detect a huge jump in temperature. However, during the time of readjustment, it is usually enough to form a large overshoot.

To overcome this issue, we implemented output limits for maintaining at different temperatures. The default output limit is 90 (out of 255). As the desired “maintainAt” temperature increases, the limit value is gradually raised. At or above 85°C, the limit is completely removed. This is probably one of the most important revision in our software, as it massively lowered the worst-case, instantaneous overshoot from over 15 °C to just above 3°C, a very impressive result.

2.8.2 Safety Features

Our microcontroller unit included two safety features for our system. The first feature is the overheat protection. Our software calls a safety check method in the beginning of heat() and cool() functions. Upon detecting temperature over 100°C, the device will enter a halt state, which shuts down the Peltier modules and the water pump and stays in an infinite loop. After checking and fixing component failures, the user can then reset the machine to exit the halt state. The safety check method will also check for the case where the thermocouple gets disconnected, namely, when the input is a negative value. Because the temperature of a bioengineering lab will never be below zero, detecting a below-zero temperature definitely means that the thermocouple was disconnected.

3 Verification

3.1 Circuit Verification

3.1.1 PWM Verification

We first verified the PWM output signal using an oscilloscope. As we swept the PWM output from 20% to 80% we saw on the oscilloscope the percentage of 5V also swept from 20% to 80%. Then we used the Equation to calculate the root mean squared (RMS) voltage, which was the 5V multiplied by the duty cycle percentage. Figure 7 below shows the verification of our PWM signal at 50% duty cycle.

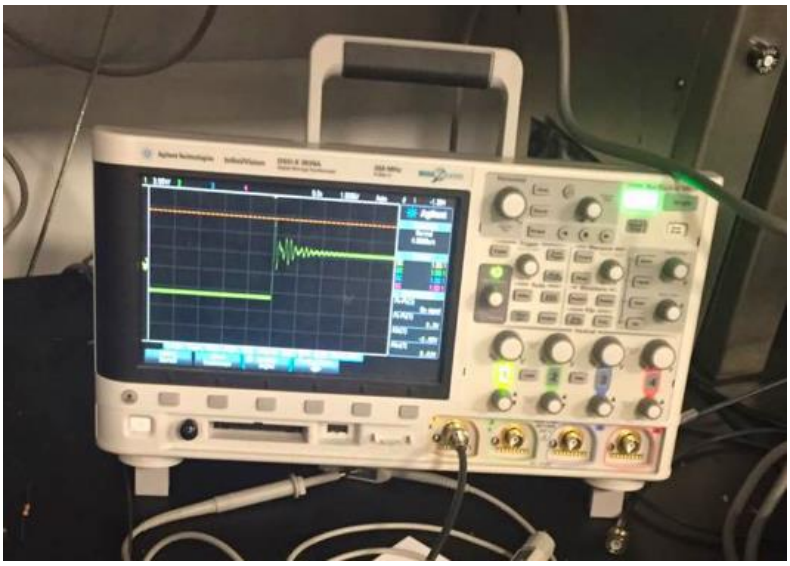


Figure 7 50% PWM Duty Cycle

3.1.2 MAX6675 verification

The operations of the MAX6675 chip was verified through the signal waveform. The microcontroller sent a sequence of clock signals to request data from the A/D converter, and the A/D converter returned temperature value in binary form with two-digit precision. The example in Figure 8 shows the waveform generated at room temperature, and reported as 22.75°C as 010110.11. When there is no connection, microcontroller will read a flat GND signal, and report error accordingly.

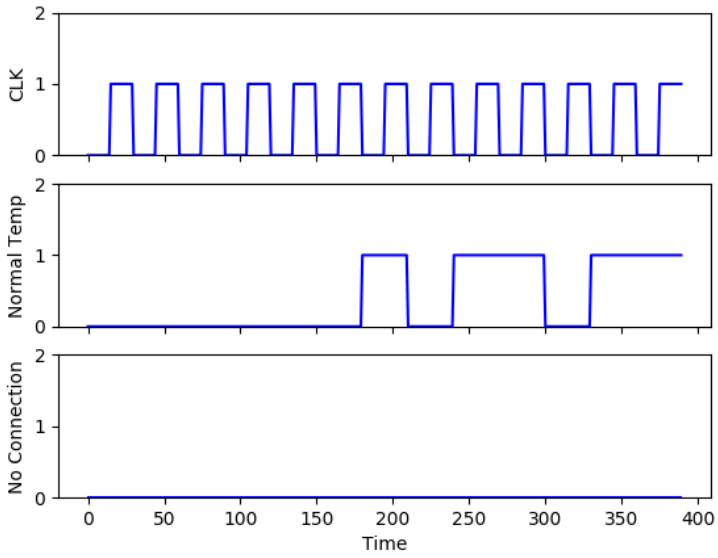


Figure 8 MAX6675 Waveform Verification

3.2 Thermocouple Module verification

While performing our PCR experiments, we verified the functionality of our thermocouple module with the IR thermometer. The error between the two readings were kept within ±5 °C. Figure 9 shows the error between the IR thermometer reading and thermocouple reading was 3 °C at around 85 °C.

Figure 9 Thermalcouple Accuracy Verification

3.3 Mechanical Verification

To verify our mechanical design, we needed to assemble all parts together. As shown in Figure 10, Peltier modules and water blocks securely fitted in the holder with minimal empty space. Wires and vinyl tubes were able to pass by without obstruction. The power supply's electrical contact points were all covered to avoid potential electrical risks. In addition, to verify that there was no leakage for the water pump module, slices of paper towel are wrapped around the water tank and tubing connections while the pump runs at maximum capacity for 20 minutes. After this time, we inspected the paper towel slices for signs of water and no evidence of water stain was found. The tubes fit around the openings of water blocks very tightly, and there is no need to design additional tube fittings.

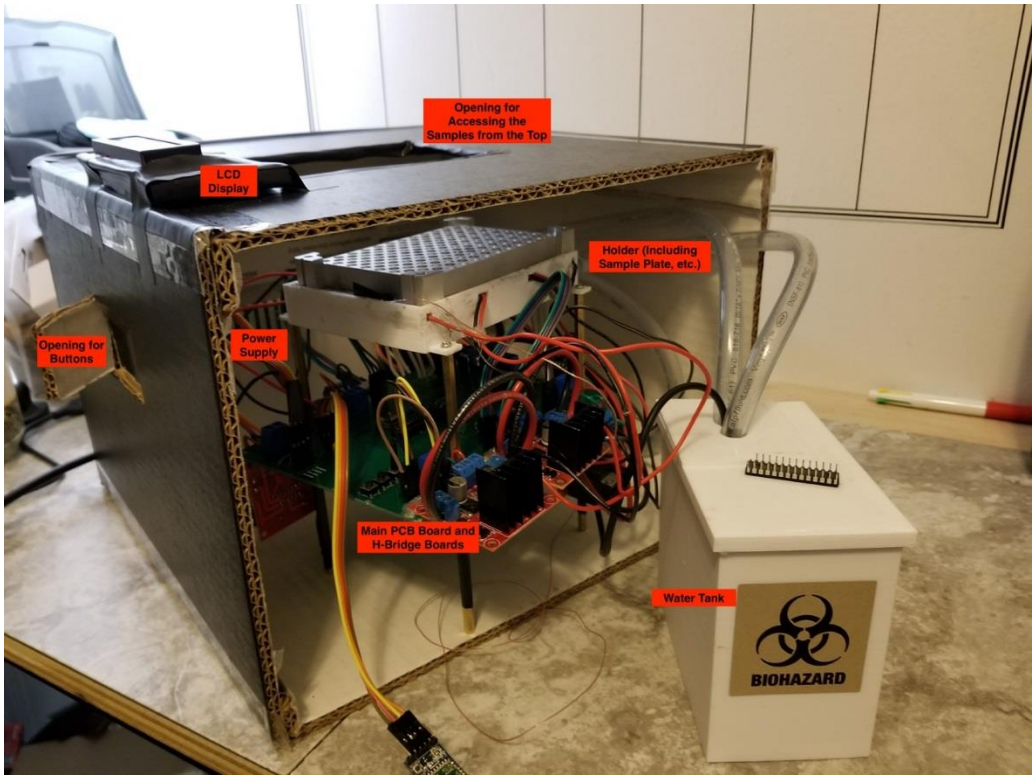


Figure 10 Finished Product (one side of casing removed for viewing)

3.4 Device Overall Functionality Verification

Because every time heatTo(), coolTo(), maintainAt() are called, the function prints necessary information to the console with a one second interval. During a procedure, such information will be continuously printed. Then, it is easy to export the entire console log to generate a .txt file and run a charting program in Python that handles this particular format of input.

We were able to use our device to perform 30 cycles of PCR operation successfully. Figure 11 shows the graph of temperature vs time over one cycle of PCR operation. The maximum overshoot and fluctuation were kept at around 2°C when maintaining the temperature, which was a huge step up from the proposed level of accuracy of ±5°C.

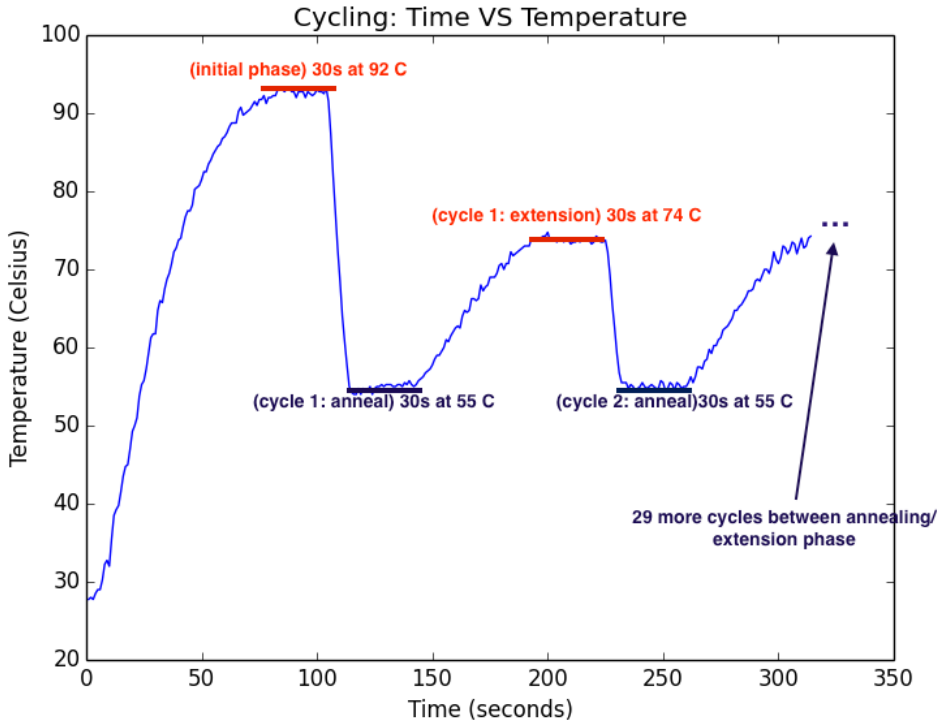


Figure 11 Temperature vs Time

4 Cost

A complete breakdown of cost of this project is listed in Table 2. The cost for customized parts by Machine Shop is based on an estimation of the unit price of the material.

We found the average annual salary for each group member is $78,500 based on starting salary data provided by Engineering Career Services. There are 365*5/7-25 = 235 working days per year, taking weekends into account and factoring out 25 days for holidays/vacations. The average hourly wage for each of our group member is roughly $78,500/(235 days)/(8 h/day) = $42/h. We spent an average of 10 hours of group working time per week for 15 weeks. Then we use the following formula to calculate the total labor cost in Eqn. 2.

$$Labor\ cost = hourly\ rate \times total\ hours \times 2.5 \quad \text{(Eqn. 2)}$$

Table 2 Cost Analysis

PARTS				
Part Name	**Manufacturer**	**Unit Cost**	**Quantity**	**Total Cost**
ATmega328 Microcontroller	Atmel	$24.95	1	$24.95
28 Pin IC socket	Uxcell	$0.62	1	$0.62
Screw Connector	Uxcell	$0.29	9	$2.61
Crystal Oscillator	Uxcell	$0.51	1	$0.51
Resistor	N/A	$0.02	7	$0.14
Capacitor	N/A	$0.09	3	$0.27
Standoff Spacer	Uxcell	$0.47	12	$5.64
PCB	PCBWay	$18.00	1	$18.00
			Microcontroller/PCB	**$34.74**
K-Type Thermocouple	Perfect Prime	$8.99	2	$17.98
MAX6675 SPI Interface	AiDeepen	$6.99	1	$6.99
			Sensor	**$24.97**
L298N H Bridge	STMicroelectronics	$6.79	6	$40.74
TEC1-12715 Peltier Cooler	MOOPACK	$6.50	6	$39.00
IRF520 MOSFET	Vishay Siliconix	$1.06	1	$1.06
Brushless Pump DC	EnPoint	$10.89	1	$10.89
Vinyl Tubing 5/16 ID	Learn To Brew	$6.00	1	$6.00
Aluminum Water Cooling Block	BXQINLENX	$8.00	2	$16.00
			Heating/Cooling	**$113.69**
Button	N/A	$0.56	3	$1.68
1602 Serial LCD Display	SunFounderDirect	$9.99	1	$9.99
SH-HC-08 BLE Slave Module	DESHENGDA	$7.99	1	$7.99

Table 2 Cost Analysis (contd.)

			User Interface	**$19.66**
12V 30A Power Supply	SUPERNIGHT	$21.49	1	$21.49
			Power	**$21.49**
Aluminum Slab	Machine Shop	$30.00	1	$30.00
			Miscellaneous	**$30.00**
			PARTS TOTAL	**$244.55**

LABOR				
Team Member	**Hourly Rate**	**Total Hours**	**Expense Multiplier**	**Total Cost**
Shaoyu	$42	150	2.5	$15750
Pei	$42	150	2.5	$15750
Yuanjiu	$42	150	2.5	$15750
			LABOR TOTAL	**$47250**

GRAND TOTAL	**$47494.55**

5 Conclusion

5.1 Accomplishments

Our team successfully developed a reliable PCR machine that is capable of performing 30 cycles of PCR experiments. We implemented all design requirements including heating and cooling system design, Control algorithm design, user interface design, and mechanical system design. We used linear regression calibration method and achieved 5°C accuracy in temperature control. Further, we also managed to improve our PID control to limit the temperature overshoot within 2°C, which is much less than the required tolerance.

5.2 Uncertainties

At this point, our group is certain that there is barely any uncertainties around our project. Everything is functional and the system has shown robustness in its operations. The product was put to a series of tests during the demo week and did not show any signs of unreliability.

5.3 Future Work

We will mainly focus on increasing the accuracy and stability of our system. Although our product has a reliable performance as a prototype, there are still many features that can be improved. In the future, we will calibrate our thermocouple to a higher precision and mount multiple thermocouples on different areas of the sample plate to minimize measurement errors. The current PID algorithm implementation ensures temperature overshoot within 2°C, but it can be further optimized for even less overshoot. Additionally, we will make clusters of PCR machines sharing a same cooling unit to increase the total system through put and reduce the cost per PCR machine.

Finally, we will setup a backend real-time monitoring system through Amazon AWS to record all experiments run and their temperature. In this way, user can compare the runs of different cycles and choose the most desired result.

5.4 Ethical Considerations

To ensure the safety of the operator, as well as to comply with the IEEE [6] and ACM Code

of Ethics [7], we have added code to handle situations that could potentially be harmful. Namely, situations where 1) thermocouple's signal is lost and 2) temperature reaches above 100°C. The temperature control software can successfully identify those potentially hazardous situations to shut down the device to prevent further damage.

Finally, we need to consider biosafety for our end user because they may place biohazardous material on and around the device. Even though it is the user's responsibility to prevent harms, there should be a biohazard sign on the device to meet ACM Code of Ethics and Professional Conduct, #1.2. [7] Our product is designed to operate at Biosafety Level 1. [8]

References

[1]Wikipedia. (2017). *Polymerase Chain Reaction*. [Online]. Available: https://en.wikipedia.org/wiki/Polymerase_chain_reaction. [Accessed: 20 Sept. 2017].

[2]NXP. (2014). *Parallel Configurations of H-bridges*. [Online]. Available: https://www.nxp.com/docs/en/application-note/AN4833.pdf. [Accessed: 1 Dec. 2017]

[3]Arduino. (2016). *Control Water Pump.* [Online]. Available: https://forum.arduino.cc/index.php?topic=406340.0. [Accessed: 8 Nov. 2017]

[4]Wikipedia. (2017). *PID Controller.* [Online]. Available: https://en.wikipedia.org/wiki/PID_controller. [Accessed: 28 Oct. 2017].

[5]Arduino. (2017). *PID Library.* [Online]. Available: http://playground.arduino.cc/Code/PIDLibrary. [Accessed: 30 Oct. 2017]

[6]Ieee.org. (2017). *IEEE Code of Ethics*. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: 20 Oct. 2017]

[7]Acm.org. (1992). *ACM Code of Ethics and Professional Conduct*. [Online] Available at: https://www.acm.org/about-acm/acm-code-of-ethics-and-professional-conduct [Accessed 20 Oct. 2017].

[8]McLeod, V. (2010). Biosafety Levels 1, 2, 3 & 4 | Lab Manager . [Online] Lab Manager. Available at: http://www.labmanager.com/lab-health-and-safety/2010/12/biosafety-levels-1-2-3-4#. [Accessed 21 Sept. 2017].

[9]MBED. (2015). L298N Breakout Test. [Online]. Available: https://os.mbed.com/teams/TVZ-Mechatronics-Team/code/L298N-Breakout-Test/. [Accessed 37 Nov. 2017].

Appendix A Requirement and Verification Table

Table 3 Full Requirements and Verification Table

Microcontroller			
Requirement	**Verification**	**Points**	**Verified**
Requirement 1: Thermocouple connection check Microcontroller will report error code if the thermocouple is disconnected	**Verification 1:** **Equipment:** N/A **Procedure:** a. Connect the device to wall outlet b. Disconnect the thermocouple from the main PCB **Expected results:** A warning message will appear on the app; the Peltier modules will stop operating, and the device will require a reboot	**5**	**Y**
Requirement 2: Button functionality When buttons are pressed, microcontroller will enter programmed procedure	**Verification 2:** **Equipment:** N/A **Procedure:** a. Press the first button to circulate through menu items b. Press the second button to confirm the selection and enter the procedure c. Press the third button to soft reboot the machine **Expected results:** Perform each operation above accordingly	**5**	**Y**
Requirement 3: Frequency tolerance Because we will use real-time control algorithm, the nature of such algorithm requires clock frequency of microcontroller to be above 8MHz	**Verification 3:** **Equipment:** Timer **Procedure:** a. Connect microcontroller to power source and flash it with test program to blink LED b. Make the counter inside the program to count clock cycles and toggle every 1 second c. Watch LED blinking **Expected results:** In 10 ± 1 seconds it will blink 5 times.	**4**	**Y**
Requirement 4: PWM duty cycle change	**Verification 4:** **Equipment:** Oscilloscope **Procedure:**	**5**	**Y**

Microcontroller must be able to send PWM with different duty cycles to power water pump at different speed.	a. Connect microcontroller to power source b. Probe output signal pin using oscilloscope c. Watch the output on oscilloscope as PWM duty cycle sweep from 0% to 100% **Expected results:** Oscilloscope will display output accordingly.		
Requirement 5: Circuit protection Microcontroller will report error code if temperature reading from thermocouple goes beyond 100°C	**Verification 5:** **Equipment:** Hot air gun blower **Procedure:** a. While the device is running a procedure, set the solder blower heat gun at 100°C or more b. Point the tip of the gun at the tip of thermocouple **Expected results:** A warning message will appear on the app; the device will enter a loop with no heating output and needs a manual reboot	**5**	**Y**
Requirement 6: PWM duty cycle calculation display Microcontroller will calculate the PWM duty cycle input to Peltier according to the PID algorithm. The value of PWM duty cycle will be displayed every one second on the app.	**Verification 6:** **Equipment:** N/A **Procedure:** a. Start the experiment procedure to let the device heat up or cool down to target temperatures. b. Observe if the app displays the PWM duty cycle value. **Expected results:** The app will display values of PWM duty cycle every one second.	**3**	**Y**

Peltier Module			
Requirement	**Verification**	**Points**	**Verified**
Requirement 1: Max temperature The Peltier module should be able to raise the temperature to 90°C	**Verification 1:** **Equipment:** IR thermometer **Procedure:** a. Measure the temperature using IR thermometer while the app is displaying "Heating to 90°C (or above)" b. Observe if the temperature exceeds 90°C	**4**	**Y**

	Expected results: IR thermometer should output 90 ± 5°C (5°C being the potential error in the IR thermometer)		
Requirement 2: Temperature control Peltier module should be able to keep microplates around constant temperature for 2 minutes with fluctuation of 5°C	**Verification 2:** **Equipment:** IR thermometer **Procedure:** a. Program the temperature to be set at 55°C because this is the annealing temperature b. Connect to power and wait 120 seconds to let temperature reach steady state c. Measure temperature then **Expected results:** Error between reads of IR thermometer and target temperature is below 5°C	**5**	**Y**

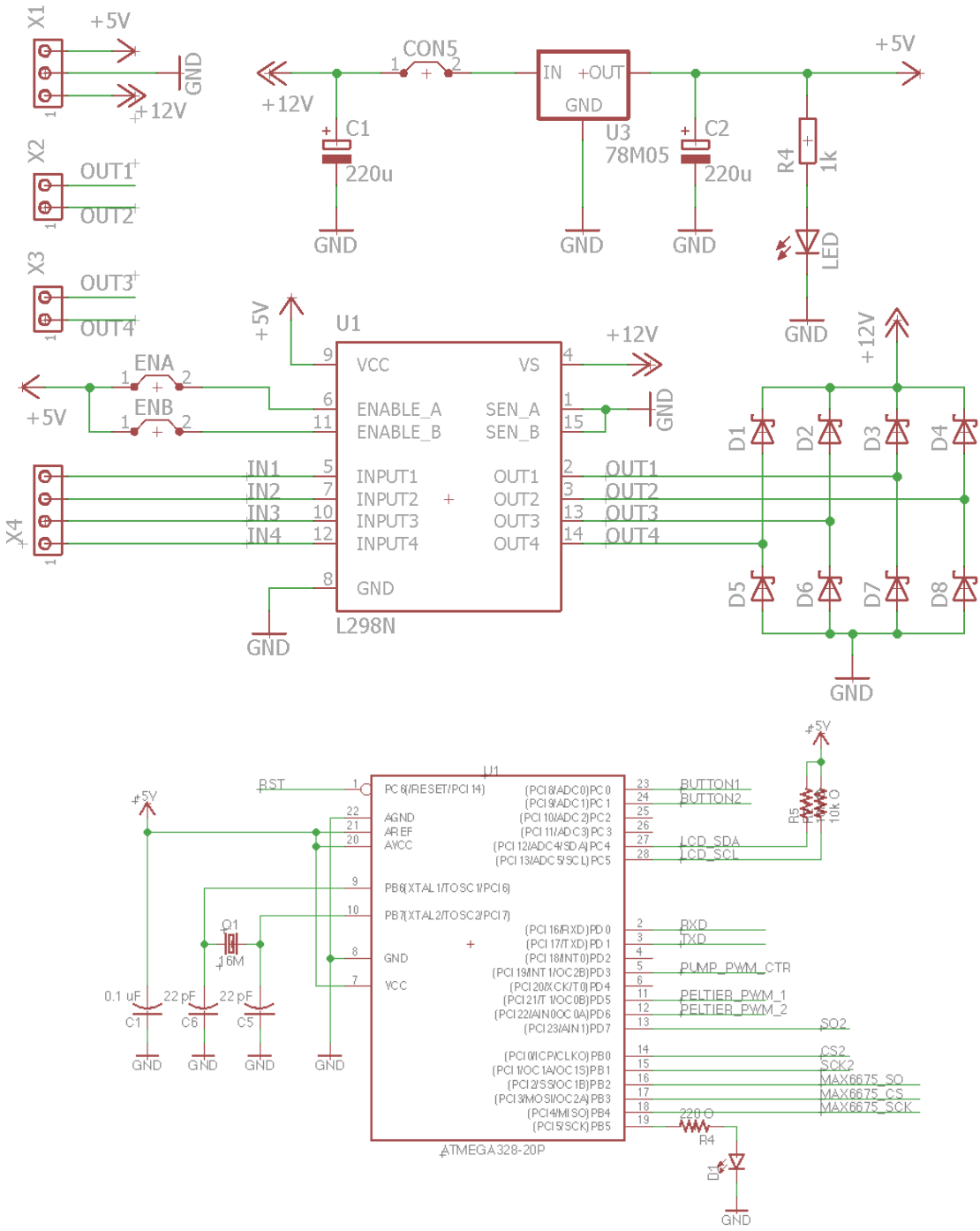
Water Pump			
Requirement	**Verification**	**Points**	**Verified**
Requirement 1: No leakage There will be no water leakage in the system	**Verification 1:** **Equipment:** Water **Procedure:** a. Power the system on b. Set the duty cycle of water pump to 80% or above **Expected results:** No water is leaked	**5**	**Y**

Thermocouple			
Requirement	**Verification**	**Points**	**Verified**
Requirement 1: Temperature sensing 1.Thermocouple will convert analog data to digital data through MAX6675, and send the digital data to MCU 2.Microcontroller will	**Verification 1:** **Equipment:** IR thermometer **Procedure:** a. Microcontroller reads digital temperature data and convert the data through a mapping table b. Display the calibrated temperature on LCD or app c. Measure the temperature with IR thermometer Expected results:	**4**	**Y**

be able to convert the thermocouple temperature data to calibrated temperature	LCD or app displays temperature information and reading error between IR thermometer and thermocouple is less than 5°C		

User Interface			
Requirement	**Verification**	**Points**	**Verified**
Requirement 1: Functionality LCD should be able to show "LifePCR" and temperature data.	**Verification 1:** **Equipment:** none **Procedure:** a. Connect microcontroller to power source b. The LCD will start showing various texts **Expected results:** "LifePCR" is displayed on LCD on boot LCD displays temperature data while operating procedures.	**5**	**Y**

Appendix B PCB Schematics



[9]

Figure 12 PCB Schematics

Appendix C Python Temperature Charting Program

```
1  import matplotlib.pyplot as plt
2
3  temperatureVal = []
4  time = []
5
6  with open("data.txt") as f:
7      for line in f:
8          if line.find("Deg")  > 0:
9              temperatureVal.append(float(line[-6:]))
10 for i in range(1, len(temperatureVal)+1):
11     time.append(i)
12         # if "Deg C = " in line
13      #      temperatureVal.append(float(line[-4:]))
14
15 # print temperatureVal
16 # print time
17 plt.plot(time, temperatureVal)
18 plt.xlabel("Time (seconds)")
19 plt.ylabel("Temperature (Celsius)")
20 plt.title("Cycling: Time VS Temperature")
21 plt.show()
```

Figure 13 Python Program to Plot Data

Appendix D Button Selection Procedure

```
// do this every 1 sec
if (millis()%1000==0) {
  /**UI**/
   if (buttonPushCounter % 3 == 1){
        Serial.println("procedure 1?");
        lcdPrintText("Procedure 1?");
    } else if (buttonPushCounter % 3 == 2) {
        Serial.println("procedure 2?");
        lcdPrintText("Procedure 2?");
    } else if (buttonPushCounter % 3 == 0){
        Serial.println("nothing selected");
        lcdPrintText("PLZ SELECT INPUT");
}

  /**END UI**/

   if (buttonPushCounter % 3 == 1 && analogRead(buttonPin2) < 512){
        lcdPrintText("Confirmed P1");
        delay(2000);
        procedure1();
    } else if (buttonPushCounter % 3 == 2 && analogRead(buttonPin2) < 512) {
        lcdPrintText("Confirmed P2");
        delay(2000);
        procedure2();
   }
```

Figure 14 Button UI Display Loop and Execution Loop