The Closed-Loop Liquid Cooling System

Prepared by:

Jayesh Gohel

Colby Hennan

Vaisnavsing Kautick

Diana Montero Baca

Sathma Sathsaree Goonathilaka

Final report submitted in partial satisfaction of the requirements for the degree of

Bachelor of Science in

Electrical and Computer Engineering

of the

University of Manitoba

Winnipeg, Manitoba, Canada

Faculty Advisors:

Dr. Arkady Major, P.Eng. - University of Manitoba

Dr. Athula Rajapakse, P.Eng. - University of Manitoba

March 2020

This page is intentionally left blank.

Abstract

In order to maintain the performance of computers, it is necessary to dissipate the heat that is generated by the CPU and other electronic components inside the computer. Liquid cooling is one of the most effective solutions to remove this heat from the microprocessors. Since the heat produced by a CPU rapidly fluctuates over time, it is referred as a dynamic load. Therefore, the main goal of this project is to cool and maintain the temperature of the CPU as close as possible to the predefined temperature by the user while considering its dynamic nature. In addition, the system can be monitored and controlled using an Android application.

This project is broken into four main components: the mechanical components, the controller subsystem, the software subsystem and the communication links. The mechanical components were assembled to ensure that the system is watertight to mitigate the risk of water damage. The controller subsystem is implemented on an Arduino Uno microcontroller connected to a TEC module and various I/O devices. The software subsystem consists of an Android application, a database, and a second Arduino Uno microcontroller which acts as a webserver.

The Arduino controller compares the user's set temperature and the temperature of the load to determine the amount of power to supply to the TEC module. Sensor data from the controller Arduino is sent to the webserver Arduino which is then transmitted to the Android application and the online database.

After testing on a static load, a load which has no temperature variations with time, the system was able to achieve 253W of cooling power with a temperature stability of ± 0.5^{o}C. The system can read the temperature of the load at a rate of 1 sample/sec, and the flow rate at <1 sample/min. Moreover, the database can store sensor data up to 72 hours. Accordingly, this project was successfully implemented and met all the deliverables on a static load. These deliverables were also met for a dynamic load except for the temperature stability which needs further testing. Due to scheduling conflicts, this testing will be done in the near future.

Acknowledgment

We would like to thank the following staff and faculty advisors who helped us throughout the year and made it possible for us to complete this final project. To our primary academic advisor, Dr. Arkady Major, for providing us with the initial idea and helping us along the way to develop this idea into a working project. To our co-advisor, Dr. Athula Rajapakse for helping us solidify our project idea. To James Dietrich, for helping us find design solutions. Finally, to Aiden for providing communications feedback in our reports, as well as our presentations.

Contributions

The table below indicates the contributions of the group members for the deliverables of the project. The **X** represents the section leader, and the **O** represents the section contributor.

Table 0-1: Team contributions table

Task	Jayesh	Sathma	Colby	Diana	Vaisnavsing
Hardware					
Sensors		O	X		
Controller		O	X		
PCB			X		
Software					
CPU temperature	X				
Database					X
Android App				X	O
Communication	O	O	O	O	X
Web Server			O		X
Integration					
Testing	X	X	X	X	X
Placement Set-up		X			
Report					
Writing the report	X	X	X	X	X

Table of Contents

List of Figures

List of Tables

Nomenclature

IoT – Internet of Thing

TEC – Thermoelectric Cooler

API – Application Programming Interface

CPU – Central Processing Unit

I/O - Input/Output

IFTTT – If-This-Then-That

LAN – Local Area Network

GUI – Graphical User Interface

PID – Proportional Integral Derivative

I^2C – Inter-Integrated Circuits

HTTP – Hyper Text Transfer Protocol

TCP\IP – Transmission Control Protocol\Internet Protocol

PWM – Pulse Width Modulation

SDA – Serial Data

SCL – Serial Clock

1 Introduction

1.1 Motivation

The performance of a computer drops if its CPU and other microprocessors are heating up while executing various complicated computer algorithms [1]. It is necessary to dissipate the heat that is being generated by these electronic components, in order to retain their efficiency and to avoid further damage to them through overheating. Liquid cooling, due to its capability of rapid heat removal, is known to be one of the best ways to remove heat from the microprocessors. Furthermore, water is widely used as the coolant in these liquid-cooling systems because of its high specific heat capacity and thermal conductivity [2].

The goal of this project was to design and implement a low power closed-loop liquid cooling system to cool a CPU to a temperature that is selected by the user via an Android application. Upon the success of this project, it is possible to improve the performance of microelectronic devices or supercomputers at a higher scale. Also, by further developing the performance characteristics of the PID controller, this system could be utilized as a viable alternative approach for applications which require lower cooling temperatures such as vaccine cooling, general water cooling for fish tanks and for an ample of industrial applications in the medical, transportation, military and defence fields.

1.2 System Overview

The closed loop liquid cooling system is a device designed to extract heat generated by static and dynamic thermal loads. A static load is one which generates a constant amount of heat over time whereas in a dynamic load the amount of heat produced fluctuates over time. For example, a hotplate is a static load and a CPU is a dynamic load.

Due to convenience and simplicity, the project was executed in four categories, namely (1) mechanical components, (2) the control system, (3) the software and (4) the communication links between the webserver Arduino, controller Arduino, the database and the Android application.

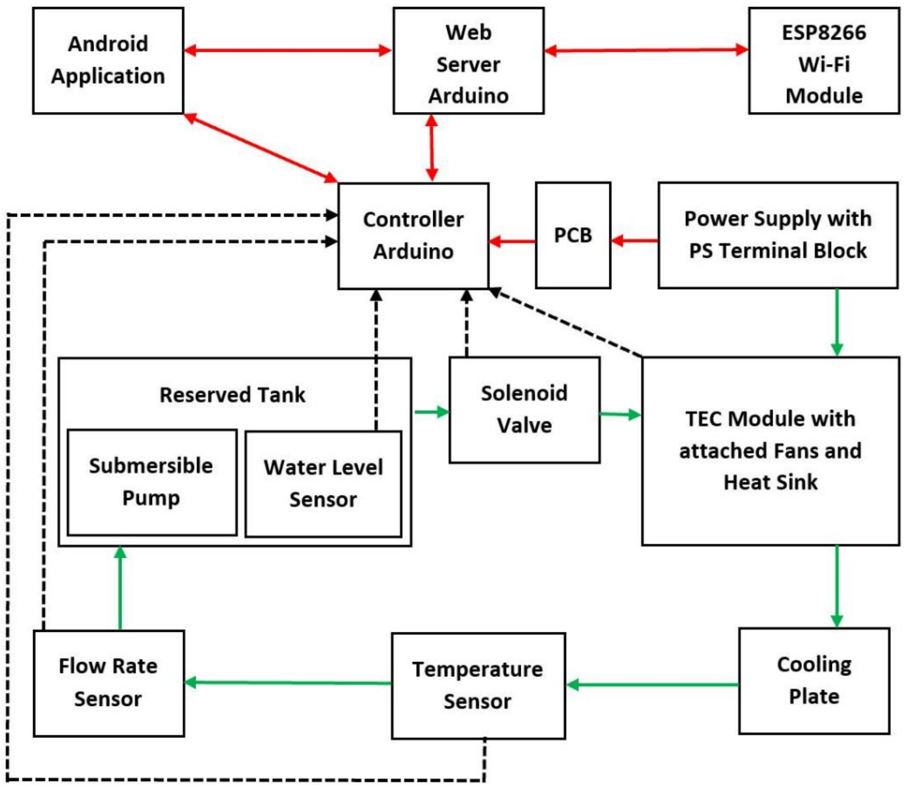


Figure 0-1: The block diagram of the complete closed loop liquid cooling system.

A block diagram describing the components of system is shown in Figure. 1-1. The mechanical parts are connected in series using a tubing of 0.9375 cm diameter, in the following order: a 1 L tank with a submerged K-0135 water level sensor, a 100 L/H water pump, a solenoid valve, TEC module with attached fans and heat sinks, a cooling plate, DS18B20 temperature sensor and a YFS201 flow rate sensor.

The controller Arduino controls the temperature of the TEC module by comparing the users set temperature with that of the process temperature, weather that be a CPU or a static load. Then it decides the amount of power to supply to the TEC module. The controller Arduino is also responsible for controlling the pump, fans, valve, and reading the flow rate sensor, water level sensor, and temperature humidity sensor. An Android application is used as a GUI where the user can monitor and control the system. An Arduino webserver is used to allow communication among all the subsystems which includes the controller Arduino, the database and the Android application. The controller Arduino sends sensor data to the webserver Arduino using I^2C communication protocol. The Arduino webserver is equipped with an ESP8266 Wi-Fi module to have a constant connection to the online database. The Android application requests data from the webserver using TCP/IP communication protocol. And lastly the Android application fetches the data that is stored on the online database, through HTTP communication. The entire system is powered by a 12 V/80 A power supply.

In addition, a leak detection system followed by an emergency shut-off solenoid valve has been integrated to the system in order to avoid damage to the electrical components of the system. The leak detection system also has an alert notification feature to send a push notification to the user via a third-party application when a leak is detected in the system or when the CPU temperature goes 5 °C above the set point temperature.

1.3 Performance metrics

The validation of the performance metrics is shown below in Table 1-1. These metrics are used to evaluate the system's overall performance including its cooling efficiency.

Table 1-1: Performance metrics comparison

Feature	Target	Outcome
Hardware		
Cooling Power	≥ 250 W	~253W (Static load)
Temperature Stability	± 0.5^{o}C	±0.5^{o}C (Static load) Not met (Dynamic Load)
Alarm trigger range (based on specified temperature)	> 5^{o}C	≥ 5^{o}C
Water Flow Rate	≤ 100 L/H	~ 60 L/H
Flow Reading Rate	1 sample/min	1 sample/min
Software		
Alarm trigger delay	< 1 s	~ 200 ms
Temperature Reading Rate	1 sample/s	1 sample/s
Data Logger Unit	≥ 48 H	≥ 48 H

1.4 Budget

The project was built successfully within our budget constrain of $500 with a total expenditure of $446.83. Table 1-2 presents the total budget of the project and a detailed listing of the parts used with their associated costs. Since it was difficult to find individual units for some components, they were purchased in a package containing multiple units. Therefore, the total cost of the project was greater than if single units were purchased. If single units were purchased, the price to mass produce the project would be approximately $424.92.

Table 1-2. Final Budget

Name/Part #	**Price Per Unit**	**Price**	**Cost to Project**
TEC Module	96.62	96.62	96.62
12V Power Supply	117.12	117.16	117.16
Flow Rate Sensor	7.73	7.73	7.73
Wi-Fi Module (2 pack)	9.90	19.79	19.79
Water Pump (4 pack)	12.64	50.56	50.56
4x Fans	37.96	37.96	37.96
Water Temperature Sensor (5 pack)	3.86	19.32	19.32
Solenoid Valve	32.69	32.69	32.69
PCB (5 pack)	13.00	65.00	65.00
Supplied Items			
2x Arduino Boards	30.00	30.00	0
Level Sensor	1.39	1.39	0
DHT11 Temperature and Humidity Sensor	1.70	1.70	0
Various Electronic Parts (resistors, push buttons, diodes, capacitors, relay, MOSFET)	10.00	10.00	0
LCD Screen	15.99	15.99	0
Water Pipe (2 meters)	3.33	19.99	0
Water tank	30.00	30.00	0
Plastic Placement Container	45.00	45.00	0
Water Resistant Foam Sheet	10.00	10.00	0
Total	**478.93**	**610.90**	**446.83**

2 Off-the-shelf Components Overview

This section discusses in details off-the-shelf hardware and software components used in this project.

2.1 Off-the-Shelf Hardware

This section details the mechanical parts taken off-the-shelf and how they were modified according to the system requirement. Off-the-shelf mechanical parts include the TEC cooling module, submersible pump, cooling plate and a water tank. The specification and usage details are discussed in their sub sections below in the order mentioned above.

2.1.1 TEC Cooling Module

In order to determine the cooling capacity and type of cooling module required for this application, it was necessary to first determine the amount of heat that needs to be extracted from various loads. Since the cooling system will be used to extract heat from a Windows core i7 CPU in the demonstration, research was conducted to determine the amount of heat produced by these processors in general. It was determined that core i7 desktop CPUs in general produce about 200 W of power [3]. Therefore, a 576 W thermoelectric Peltier cooler device was chosen for this application. It is composed of 8 TEC1-12706 Peltier elements each with a maximum cooling capacity of 60 W as shown in Figure 2-1. Each Peltier element requires 16 V and a current of 6.1 A when operating at its maximum cooling capacity. Therefore, the total cooling capacity of the entire cooling module is 480 W at ideal conditions. The Peltier element is made of two different materials connected in parallel with a semiconductor between them [4].

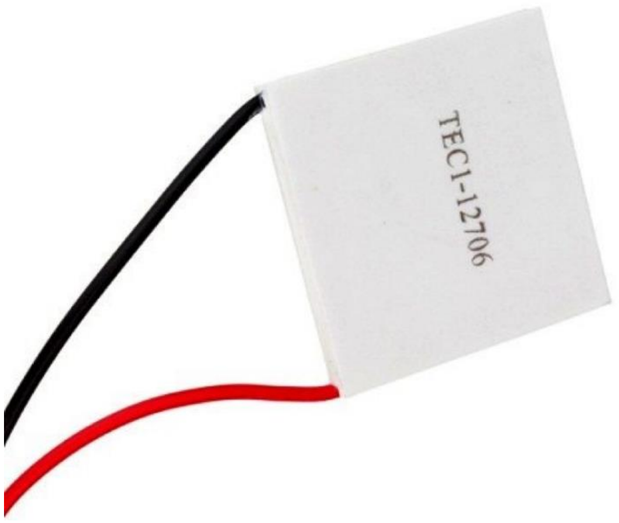


Figure 0-2: A Peltier element [5].

A cooling module with Peltier elements was chosen because they are operated by driving current through these elements. That is, when current is passed through the TEC element, one side of it gets cold while the other heats up [4]. Since they are driven using current, this can enable precise control of the cooling system. Although the cooling capacity of the TEC cooling module is 480 W under ideal conditions, the practical cooling capacity of the TEC was determined, using equation (2.1) [6], and that is, 253 W. This considers all the heat losses within the TEC module and the thermal conductivity limitations of the water-cooling block.

$$C = \frac{(\rho * V * Cp)}{t} \Delta T \qquad \begin{aligned} &C = Cooling\ Capacity\ (Watts) \\ &\rho = Water\ Density\ (\frac{\mathrm{kg}}{\mathrm{m3}}) \\ &V = Volume\ of\ Water\ (m3) \\ &Cp = Specific\ Heat\ (\frac{Joules}{kg} * Kelvin) \\ &t = time\ (seconds) \\ &\Delta T = Change\ in\ Temperature\ (Kelvin) \end{aligned} \tag{2.1}$$

Initial configuration of the cooling module was compact as shown in 2-2.

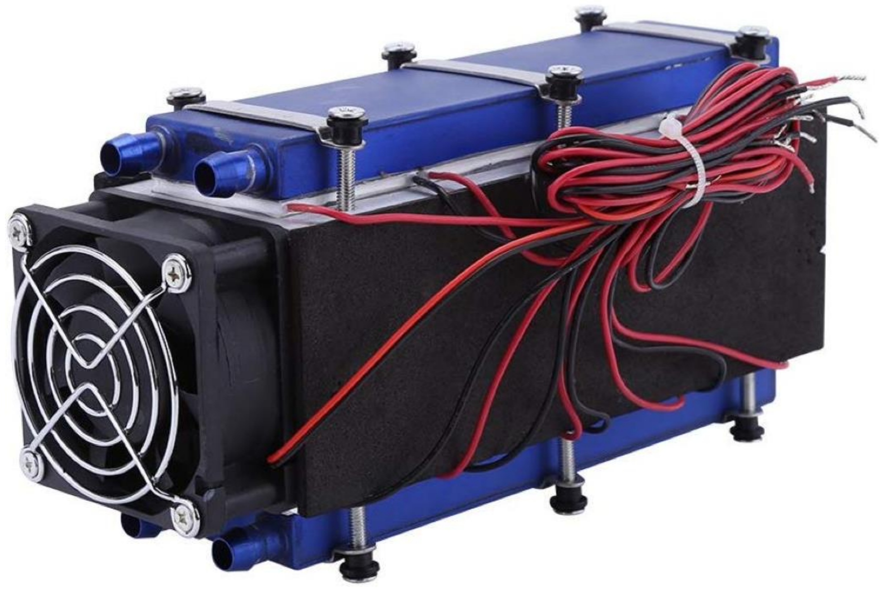


Figure 0-3: Compact cooling module [7].

Four Peltier elements are attached on each side of the heat sink as shown in Fig. 2-3. The water-cooling blocks in blue, through which the water flows, is attached to the cold side of the Peltier element while the hot side of the Peltier elements attached to the heat sink using conductive paste. The temperature of the water which is flowing through these cooling blocks decreases and the heat from the hot side of the Peltier elements is extracted and transferred to the heat sink fins through the process of conduction. Thereafter, the fans remove the heat into the ambient air via convection. Testing the compact configuration of the TEC module resulted in a problem where the TEC module initially dropped the temperature of the water from room temperature to about 14^{o} C and shortly after, the TEC module would heat up the water back to room temperature as shown in Figure 0-4: Result of a compact cooling module..

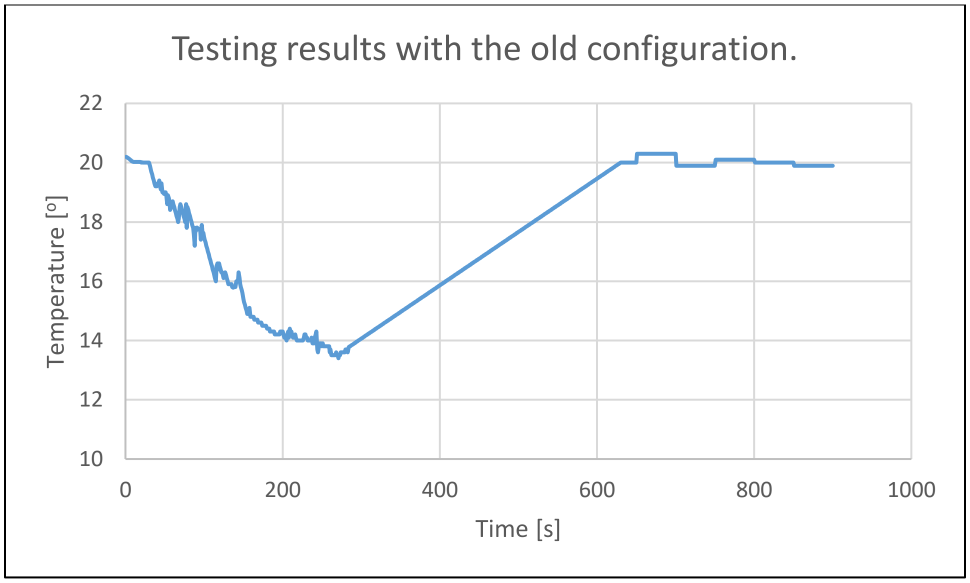
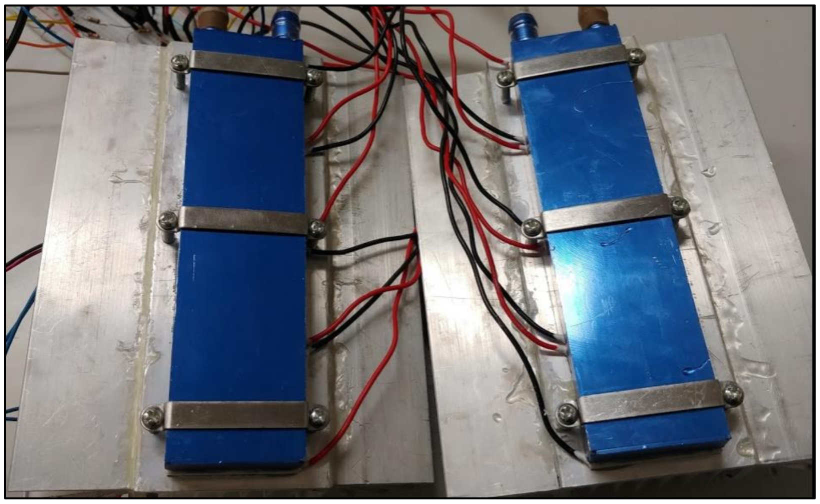


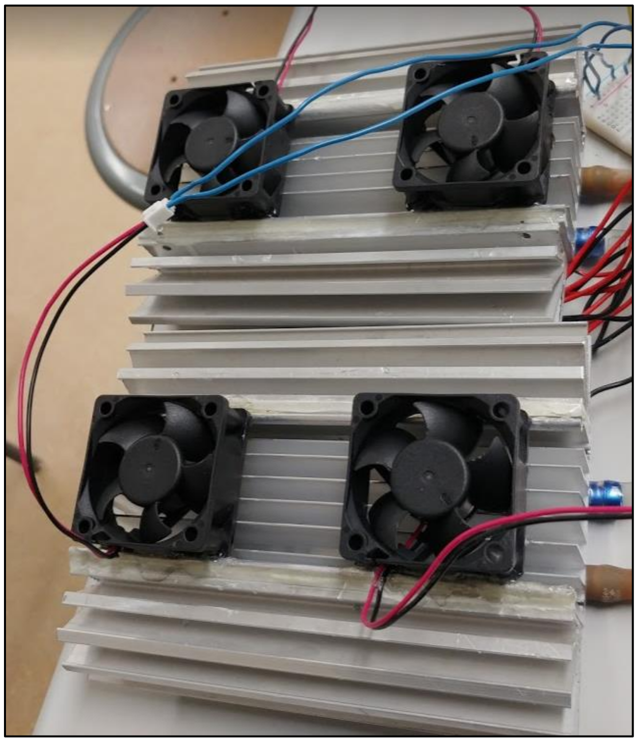
Figure 0-4: Result of a compact cooling module.

After inspection, it was discovered that the functioning of the Peltier elements are based on the temperature difference between the two sides of the Peltier elements. Reviewing the TEC1-12706 datasheet, the maximum possible temperature difference between the two sides is 70^{o} C [8]. Since the fans cannot dissipate sufficient amount of heat from the fins of the heat sinks, the temperature of the hot side of the Peltier element rises and as a result increasing the temperature of the cold side back to the room temperature, in order to maintain the temperature difference of 70^{o} C between the two sides of the Peltier element. Thus, the water flowing through the blue cooling block was brought back to initial conditions and maintained at room temperature.

The configuration of the TEC cooling module was changed from being compacted to an open configuration as shown in Figure 2-4(a) and three additional fans were attached to the fins as shown in Figure 2-5(b).



(a)



(b)

Figure 0-5: The new configuration of the TECs.

Opening the TEC cooling module and adding fans increased the heat dissipation from the heat sink fins. This allows the hot side of Peltier element to maintain a lower temperature than that of the temperature in the earlier configuration. Since the temperature difference between the hot side and cold side of the element is constant based on the amount of heat that is extracted, the temperature of the cold side is lowered by a significant margin. Thus, the temperature of the water which is running in the blue cooling blocks is maintained below the room temperature as shown in Figure 2-5.

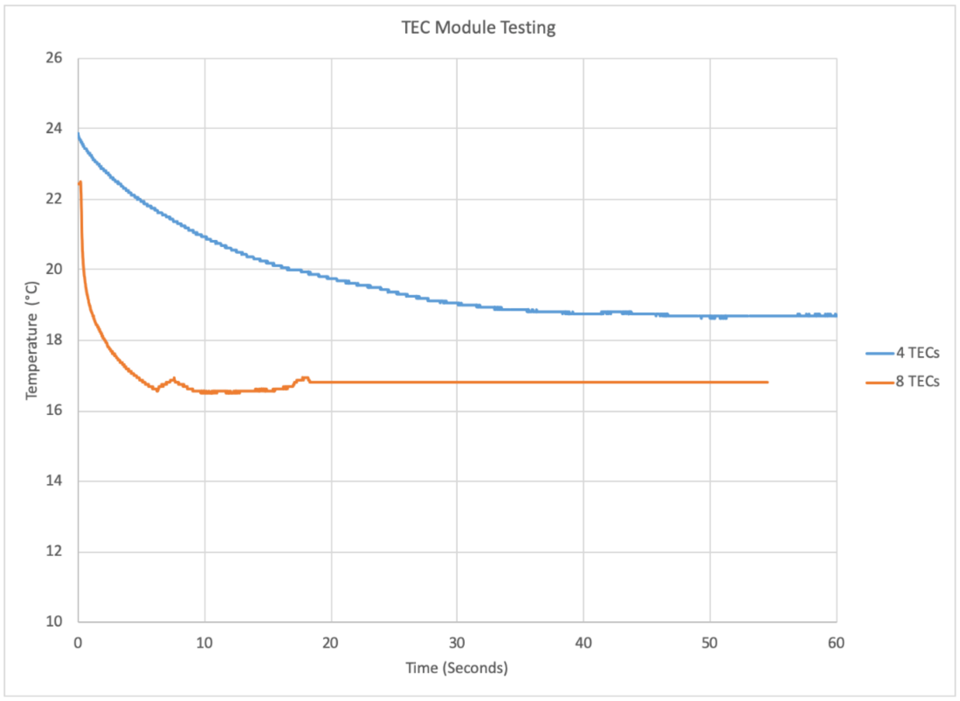


Figure 0-6: Results from new configuration of the TECs

2.1.1 Submersible Pump

A pump as shown in Figure 2-6 compatible with Arduino with a rating of approximately 100L/H was chosen for the project. The reason to have a pump with a flow rate of 100L/H is to allow ample time for heat to be extracted from the water flowing in the cooling module. Furthermore, the flow rate is fast enough to extract heat from the CPU.

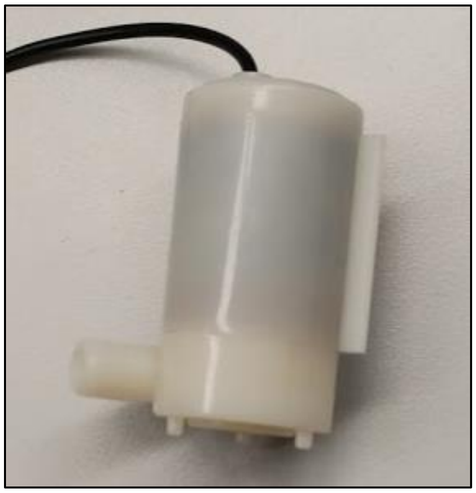


Figure 0-7: Submersible pump

2.1.2 Cooling Plate

Initially the cooling plate was to be custom designed but due to the project's time constraints and many other complications in designing its parts, it was decided to use an off-the-shelf cooling plate as shown in Figure 2-7

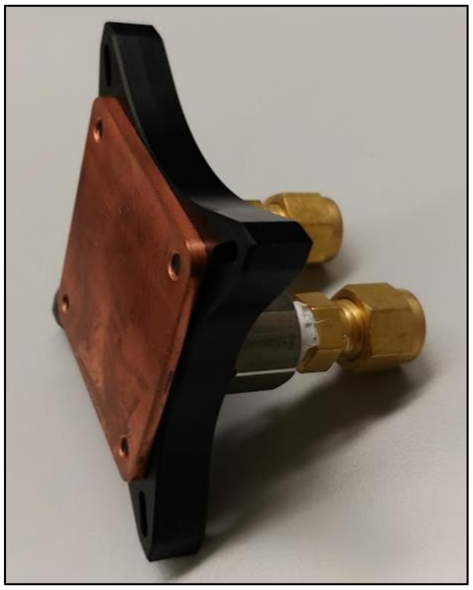


Figure 0-8: Cooling Plate

2.1.3 Water Tank

A water tank that can hold up to 1 liter of water was chosen for the project. This ensures that the heat extracted from the cooling plate is distributed over a large volume of water minimising the increase in the water temperature. Thus, the TEC cooling module will require less effort to extract the heat from the water. In addition, the tank volume of 1 L was chosen in order to minimize the size of the entire cooling system. Off-the-Shelf Software

2.2 Off the shelf Software

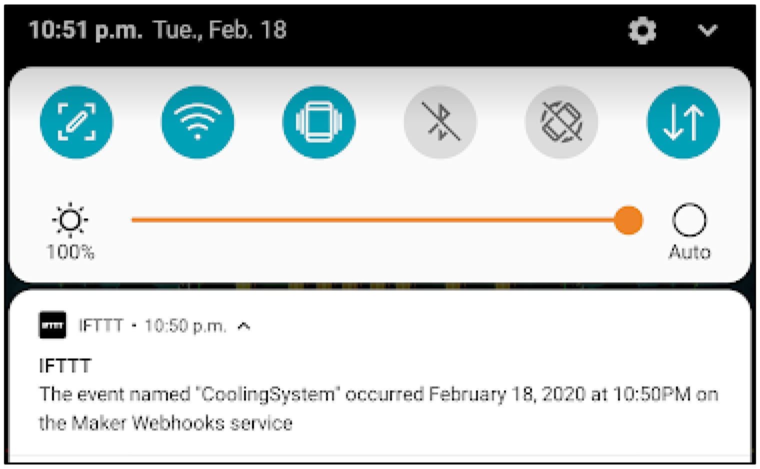
This section details the software taken off-the-shelf and its modification according to the system requirement. The off-the-shelf software include the CPUThermometer application, If-This-Then-That application and the Thingspeak database. The specification and usage details are discussed in its sub sections below in the order mentioned above.

2.2.1 CPUThermometer

The CPUThermometer is an application that is used to extract the temperature of the CPU's internal temperature sensor when the system is being used to cool a CPU. The application is compatible with Windows 10 operating system.

2.2.2 If-This-Then-That

The IFTTT is an application that has been set up to send alerts as push notifications when the controller Arduino sends a distress signal to the database. This application was chosen as its compatible with Thingspeak platform in which the database was created. Figure 2-8 shows the results IFTTT sending a push notification when a test distress signal was sent by the controller Arduino.



IFTTT
The event named "CoolingSystem" occurred February 18, 2020 at 10:50PM on the Maker Webhooks service

Figure 0-9: Alert notification.

2.2.3 Thingspeak

Thingspeak is an open–source IoT application which provides an off-the-shelf API to store and retrieve data from different sources using HTTP protocol over the internet or LAN [9]. The Thingspeak application offers a lot of off-the-shelve API's which makes it easier to work with and troubleshoot.

3 Control System

This section describes the control system of the liquid cooling system developed in the project. First, an overview of the control system is described, then the inputs and sensors are discussed according to their purposes, the reasons for selection and the calibration methods used. Next, the outputs of the project are considered. Thereafter the integration of the I/O's with the microcontroller and the PCB design process are discussed. Lastly, the temperature controllers are discussed with their associated results.

3.1 Control System Overview

An Arduino Uno is used as the microcontroller of the control system since it is powerful enough for our application, provides an adequate number of I/O pins and has PWM pins required to control the TECs. The controller Arduino serves three main purposes: first to control the TEC module, second to control the states of the output devices, and lastly to read the status of the sensor inputs. The controller Arduino uses I^{2}C protocol to communicate with the webserver, to send and receive information. This is discussed more in Section 5.

The control system consists of seven inputs namely the water temperature sensor, the flow rate sensor, the ambient temperature and humidity sensor, the water level sensor, and 2 pushbuttons. The outputs include the eight TEC elements, a water pump, four fans, a solenoid valve and an LCD screen. A system schematic of the IO devices is shown in the appendix.

3.2 Sensors and Inputs

The reason for selecting the chosen sensors, accuracy, reliability, and calibration procedure are discussed in the following sections.

3.2.1 Water Temperature Sensor

A water temperature sensor is needed for the feedback signal of the control system to make the closed-loop system. The temperature stability of ± 0.5^{o}C performance metric needed to be considered when selecting the appropriate temperature sensor. A DS18B20 waterproof temperature sensor, shown in Figure 0-10: Water temperature sensor was chosen since it is accurate within ± 0.5^{o}C, has an operating range of -55^{o}C to +125^{o}C, and is IP67 waterproof. These specifications make this temperature sensor ideal for this application. This sensor uses the Arduino

One wire and Dallas Temperature libraries to calculate the temperature. Therefore, no calibration was needed but verification was performed using a standard mercury thermometer.

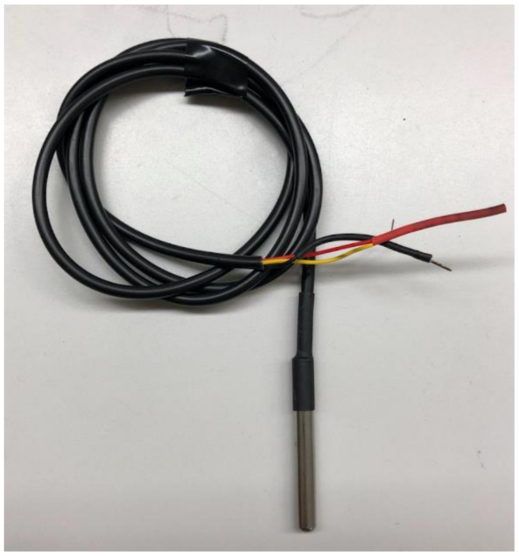


Figure 0-10: Water temperature sensor

The best way to regulate the temperature of the water is to measure the temperature of the water as close as possible to the output of the thermal load that is the cooling plate. To do this, the temperature sensor is needed to be placed in line with the water line. After searching for an off-the-shelf fitting and not finding one with the appropriate fit, it was decided that 3D printing the part was a suitable solution. Figure 0-11: Water Temperature Sensor Fitting shows the finished design of 3D printed water temperature sensor fitting.

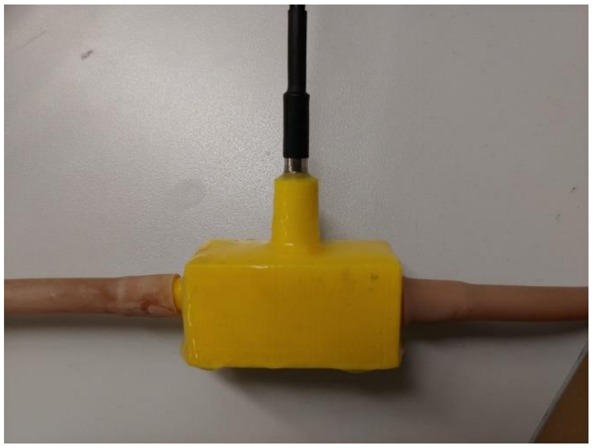


Figure 0-11: Water Temperature Sensor Fitting

3.2.2 Water Level Sensor

The water level sensor is used to detect a leak in the system and warns the user when the water tank level decreases. A Kumantech K-0135 water level sensor, as shown in Figure 3-3 was chosen since it is large enough for a 1 L tank and it integrates with the Arduino simply. The sensor was programmed so that if the water in the tank drops by 1.1 cm an alarm is triggered, sending a notification to the user through the IFTTT app.

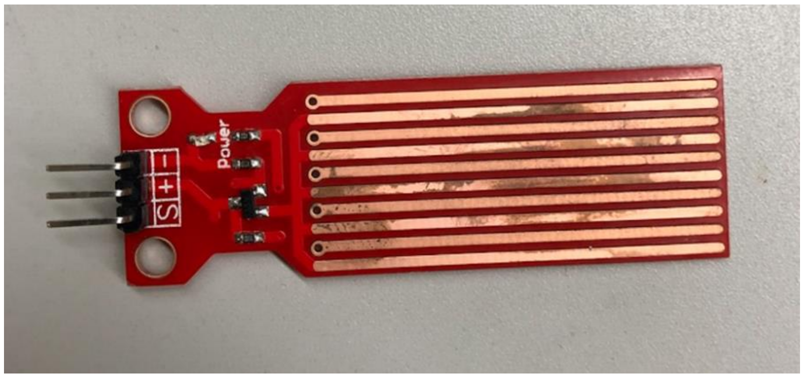


Figure 0-12: Water Level Sensor

3.2.3 Flow Rate Sensor

Initially the flow rate sensor was to be used as the main method of leak detection but after a redesign, it was decided that it will be used as redundancy for leak detection in case the water level sensor fails. The YF-S201 flow rate sensor, shown in Figure 0-13: Flow rate sensor was chosen for its low price, reliability, and ease of use with the Arduino. The sensor was calibrated manually to an accuracy of ± 1 L/H.

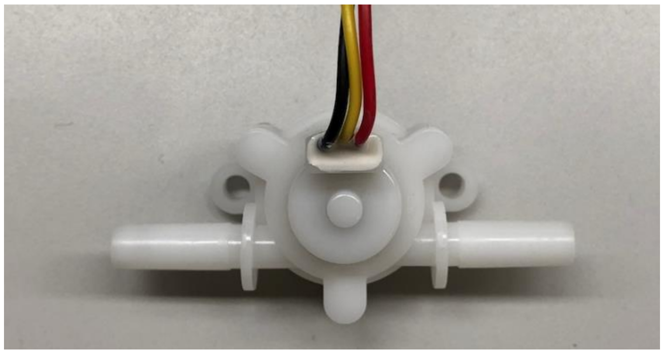


Figure 0-13: Flow rate sensor

3.2.4 Ambient Temperature and Humidity Sensor

An ambient temperature and humidity sensor was added to limit the water temperature above the dew point temperature, so that condensation does not build up within the system. An alarm will be triggered if the dew point temperature increases above the set point temperature. A DHT11 temperature and humidity sensor, show in Figure 0-14: Ambient temperature and humidity sensor, was chosen for its low price, reliability and ease of use.

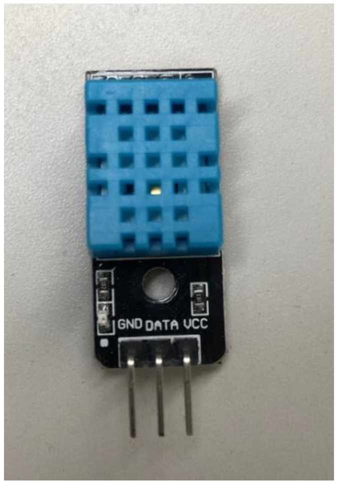


Figure 0-14: Ambient temperature and humidity sensor

3.2.5 Push Buttons and Toggle Switch

Two push buttons are used so that the user can control the set temperature from the device along with the Android application. The pushbuttons that have been chosen were the B3F tactile switch for their simple integration with the Arduino and their ergonomic design. The toggle switch will act as the on-site emergency shutoff switch for the system. When it is in the off position, the TECs, pump, and fans will all be turned off, but the parts used to monitor the system, such as the sensors and Arduinos, will continue to run in the background, and display the temperature on the Android application.

3.3 Outputs

The output of the control system consists of the eight TEC elements, a water pump, four fans, a solenoid valve, and an LCD screen. The following sections will discuss the reasons for selection, control methods and their integration with the microcontroller.

3.3.1 Peltier Elements

The TEC module selection which is composed of eight TEC1-12706 Peltier elements, is discussed in section 2.1.1. These elements are suitable for the project’s application since they will supply enough cooling power to cool a typical microprocessor in a PC. Two of these elements are placed in parallel connected to the drain of an IRLZ44N logic level MOSFET, so that it can be controlled though the Arduino’s PWM output pins. Figure 0-15: Schematic of Arduino interfaced with 2 TEC elements shows the wiring diagram of the TEC elements with the controller Arduino

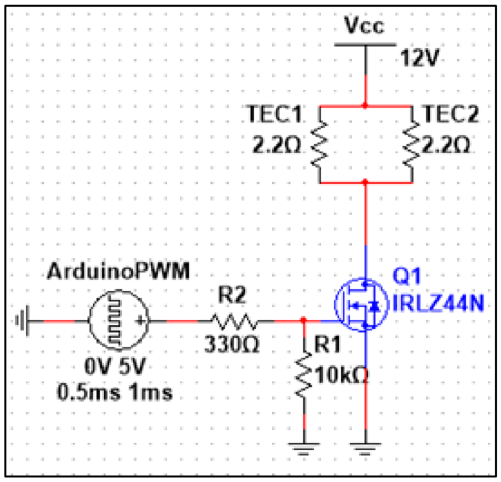


Figure 0-15: Schematic of Arduino interfaced with 2 TEC elements

3.3.2 Water Pump

One of the design constraints set out in the proposal was that the water flow rate needs to be less than 100 L/H to ensure optimal thermal transfer. A 12 V submersible mini pump, shown in Figure 0-16: Water Pump, was chosen for its compact size, reliability, and it has a rated flow rate of 100 L/H. Since the system does not need to control the flow rate of the water, a 5 V Hamlin HE3621A0500 relay can be used with the Arduino so that it can turn the pump on and off.

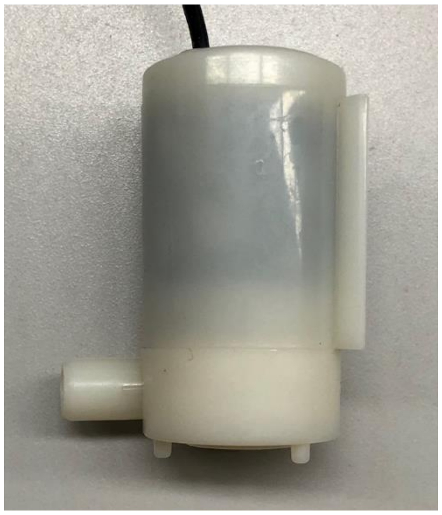


Figure 0-16: Water Pump

3.3.3 Fans

Four additional fans were added to the system to dissipate more heat from the heatsinks the reasoning behind this was discussed in section 2.1.1. It was determined that four AB5020H12 50mm x 50mm fans, shown in Figure 0-17: Fan AB5020H12, would be suitable for the project since they are cheap, powerful and fit perfectly onto the heatsinks. Two fans are wired in parallel and then connected to a 5 V relay so that the Arduino can turn them on and off.

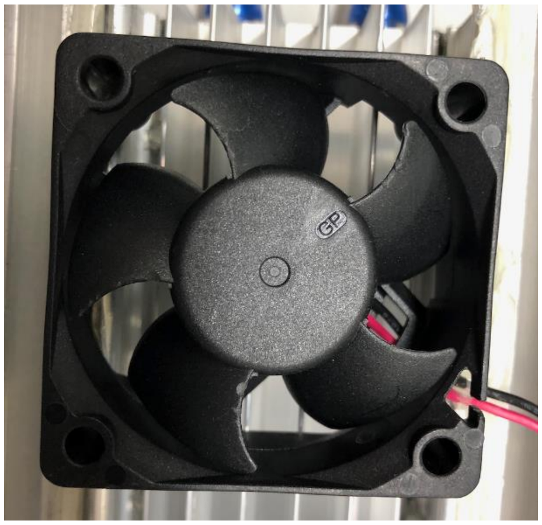


Figure 0-17: Fan AB5020H12

3.3.4 Solenoid Valve

A valve is needed in case a leak is detected to ensure the PC or application is not at risk of become damaged from water. A 12 V solenoid valve, show in Figure 0-18: Solenoid valve., was chosen for its small size, low power and low cost. This valve is interfaced through a 5V HE3621A0500 relay with a switching voltage of 200V to the Arduino so that the latter can open and close the valve.

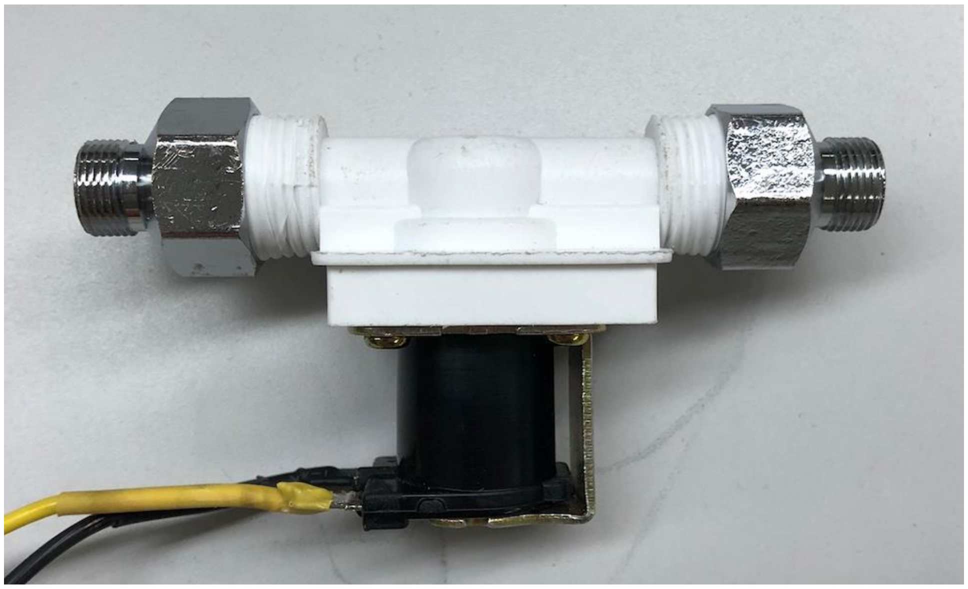


Figure 0-18: Solenoid valve.

3.3.5 LCD Screen

The purpose of the LCD screen is to show the operation of the system when the user is not using the Android application. Since the LCD is not required to show a lot of information, a Canaduino LCD 1602 2x16 LCD was selected, shown in Figure 0-19: Canaduino LCD screen.. This screen uses the I^2C protocol which reduces the number of pins used on the Arduino.

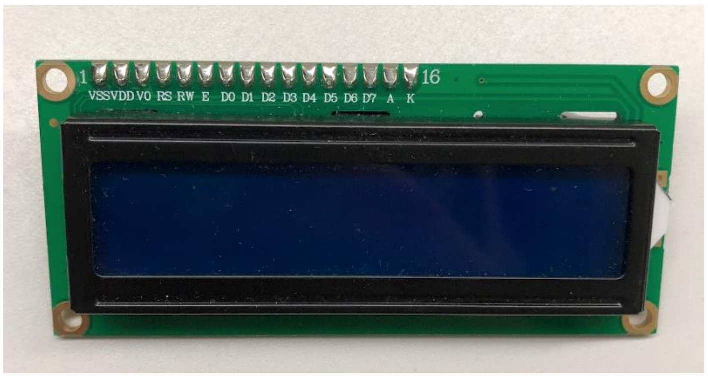


Figure 0-19: Canaduino LCD screen.

3.4 PCB Design

A stretch goal of designing a PCB for the connections between the Arduino and the I/O was added shortly after design review 2. This would allow the physical electronics of the system to be more organized, reliable, less error prone, and easier to troubleshoot.

KiCad was the software used to design the PCB considering its simplicity, large number of features, and at no cost. All of the I/O's of the system can be interfaced with simple through hole components such as resistors, capacitors, diodes, MOSFETs, and relays. As none of the group members had ever designed a PCB earlier, it was decided that a double-layer through-hole PCB would be sufficient for the project. Since the Peltier elements require a lot of power and current, the trace thickness of the PCB needed to be taken into consideration, for these traces a thickness of 3.5 mm was used as calculated from an online PCB trace thickness calculator [10]. Figure 3-11 shows the design of the PCB and Figure 3-12 shows the 3D model of the PCB.

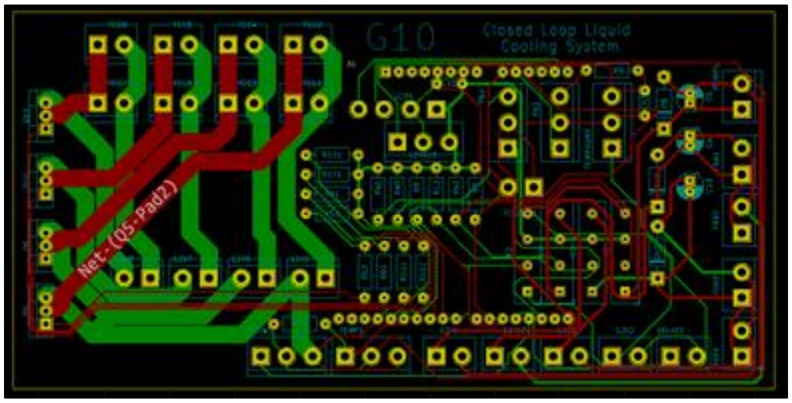


Figure 0-20: PCB design

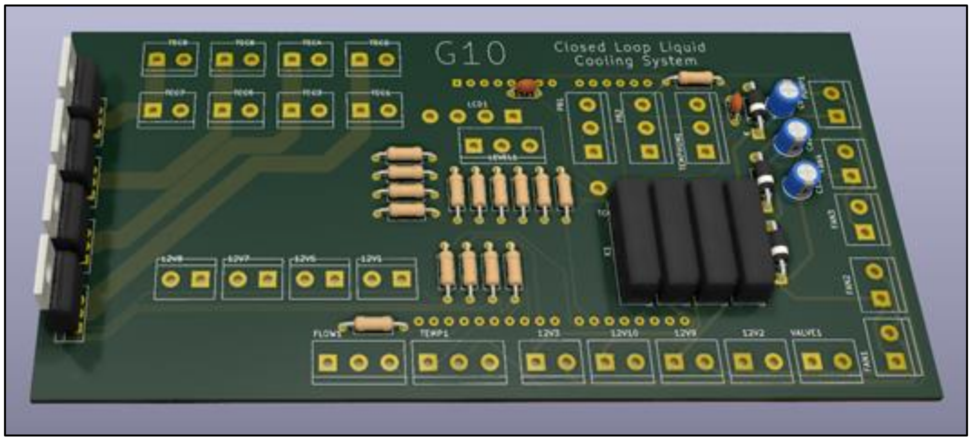


Figure 0-21: 3D simulation of the PCB

3.5 Temperature Controllers

Two temperature controllers are available in this cooling device for the user to select, depending on the application of the system. The On/Off controller can be used for light loads or applications that do not require high performance while the PID controller can be used for high performance applications. The following section will discuss how these two controllers work and their performance results.

3.5.1 On Off Controller

An on/off controller is the simplest form of temperature controller. It works by having the user to set a temperature for the controller to read. If the temperature is above the set point, the controller will turn on the TEC elements, since we want to lower that temperature. If the temperature goes below the users set temperature, then the controller will shut off the TECs and this cycle continues. This results in a continuous fluctuation in temperature around the user's set point. A flow diagram of an on off controller is shown in **Error! Reference source not found.**-1 3.

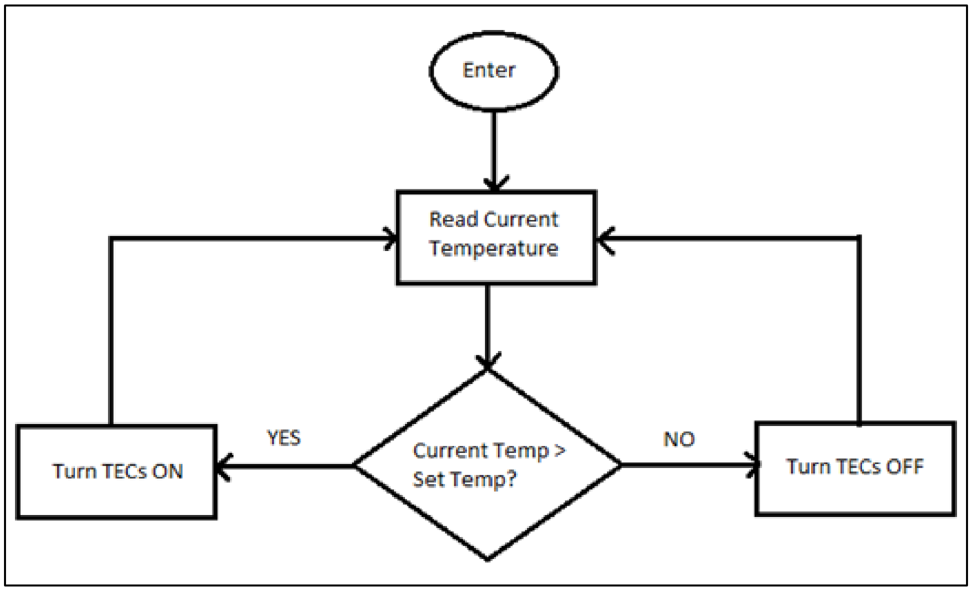


Figure 0-22: Flow chart of controller

Figure 0-23 shows the results of an on/off controller with no load on the system. Furthermore, it shows continuous oscillations around 20^{o} C with a maximum deviation of 0.37^{o}C and a minimum of -0.5^{o} C. Therefore, this controller does regulate the water temperature within ± 0.5^{o}C and it meets the temperature stability performance metric.

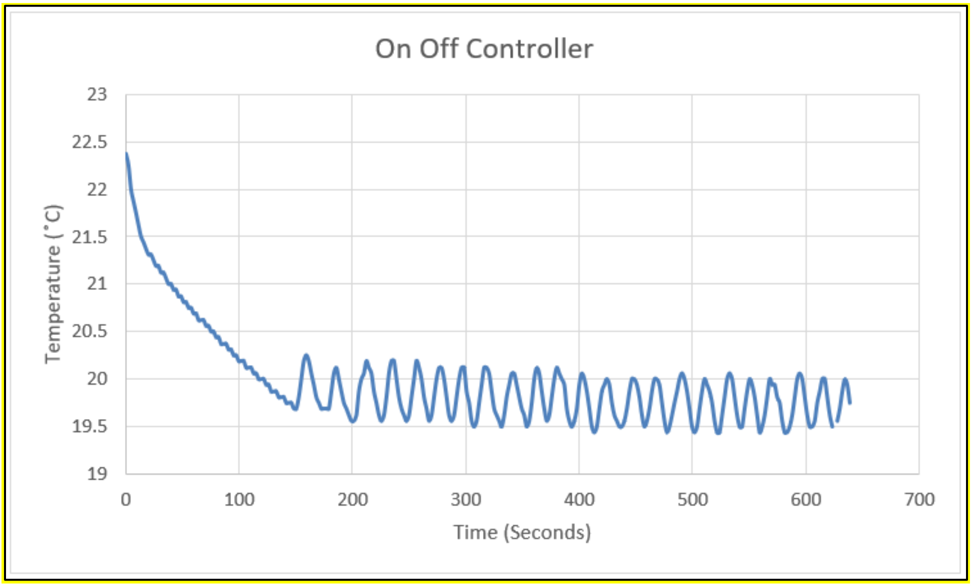


Figure 0-23: On Off Controller Response

3.5.2 PID Controller

PID controller is a popular controller used for temperature control systems to improve the closed-loop performance of the system. It uses the difference between the user set temperature and the current temperature multiplied by a PID algorithm to predict the amount of power that is needed at the output of the controller. This algorithm ensures that the system temperature remains as close as possible to the set point [11].

The three elements to a PID algorithm are the proportional, integral and derivative. The proportional is the difference between the set temperature and the current temperature multiplied by the proportional gain. The integral term is the sum of error over time multiplied by the integral gain. The derivative term is calculated by determining the rate of change of the error over time multiplied by the derivative gain. The PID gains can be tuned such that desired results are achieved [12]. The block diagram shown in Figure 0-24 describes the closed loop cooling system with a PID controller implemented.

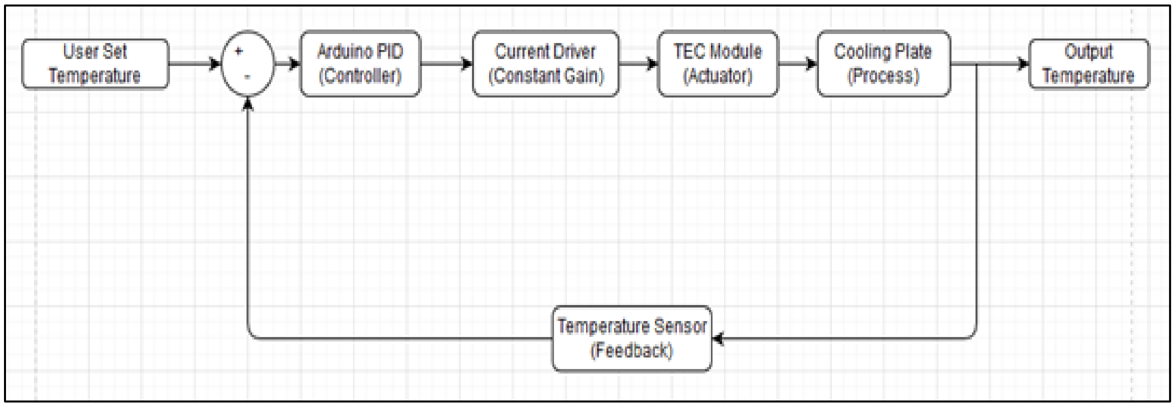


Figure 0-24: Block diagram of closed loop liquid cooling system with PID controller

A PID controller was implemented on the Arduino Uno and the parameters were then tuned to increase the stability of the temperature in order to get the most desirable performance of the system. Figure 0-25: Response of various PID controllers shows the results of implementing a PI controller on the system with a static load and various PID parameters. Increasing the value of K_p will result in a faster settling time but if K_p becomes too large the system becomes less stable. K_i also affects the settling time of the system but does not have a significant effect on the percentage overshoot or stability. The results from tuning the controller concluded that the best results were achieved with values of $K_p = 550$ $K_i = 1$ $K_d = 0$. Figure 3-17 shows the plot of the system when Kp=550 Ki=1Kd=0 zoomed in with a set point of 20 ^{o}C. From analyzing the plot, it was determined that the temperature stability with a PID controller implemented resulted in a temperature stability of $\pm$ 0.3^{o}C

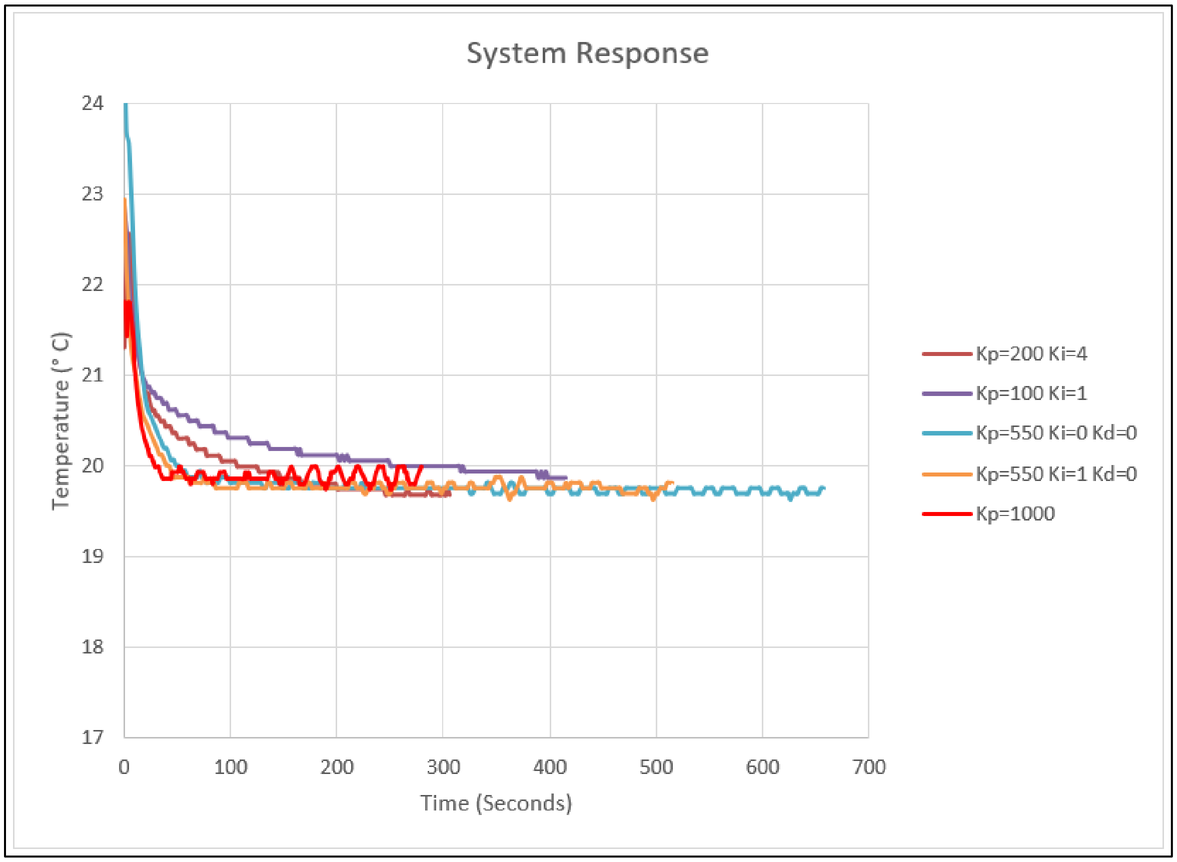


Figure 0-25: Response of various PID controllers

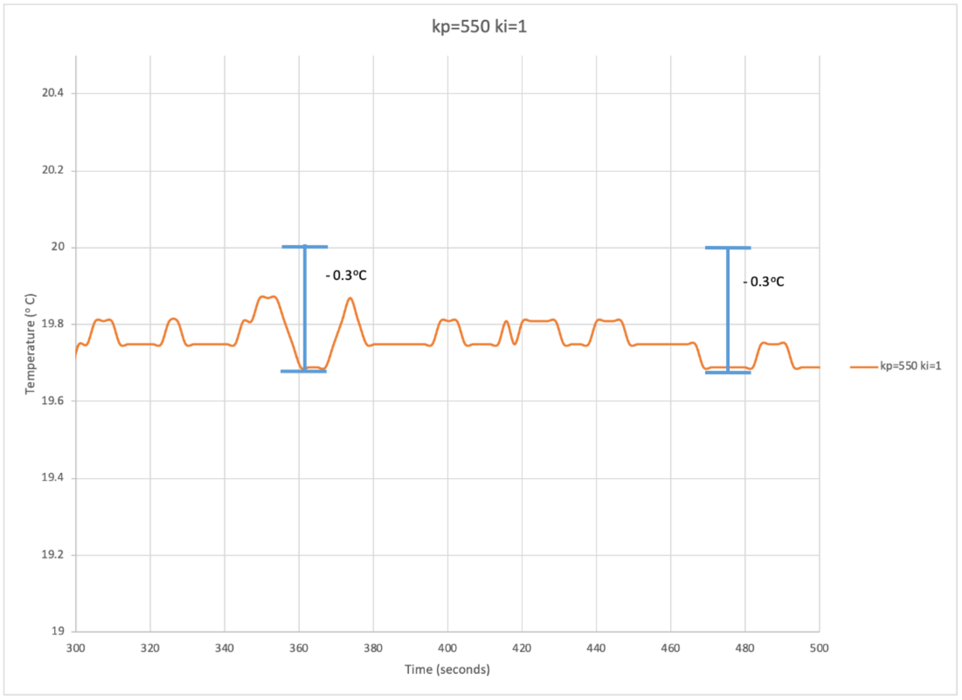


Figure 0-26: Plot of the system when Kp=550 Ki=1Kd=0 with a set point of 20 ^{o}C

4 Software

The software system consists of three main components: the Android application, the Thingspeak database and the Arduino webserver. As shown in Figure 4-1, all the components are interconnected wirelessly and the communication between the app, the webserver Arduino and the database are half duplex, which means that the data can flow in each direction simultaneously. However, the communication between the webserver Arduino and the database is simplex which means that data flows only in one direction, from the webserver to the database. Each of the components and their associated communication links are discussed in more details in the following sub-section of this report.

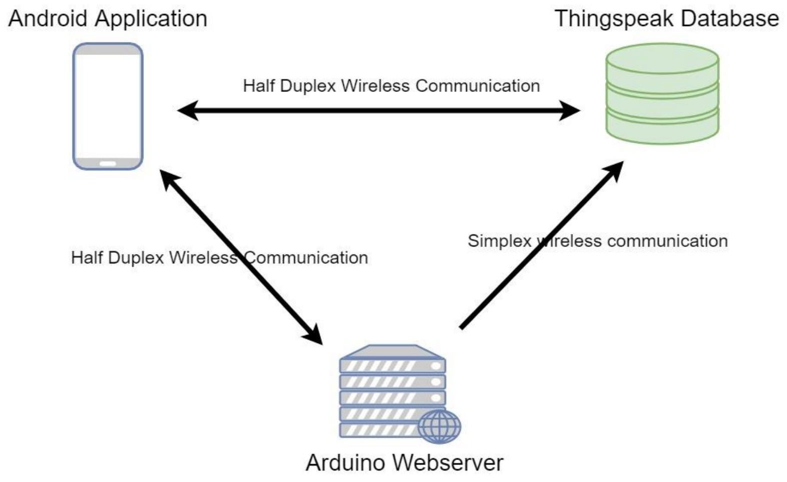
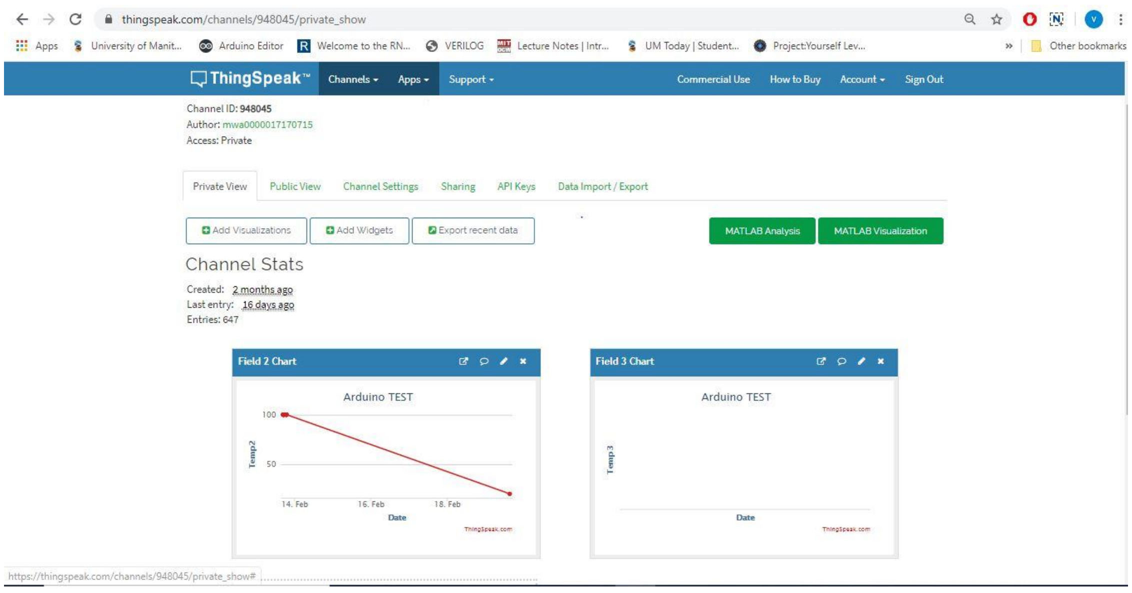

Android Application
Thingspeak Database
Half Duplex Wireless Communication
Half Duplex Wireless Communication
Simplex wireless communication
Arduino Webserver

Figure 0-27: Overview of the software aspects of the microprocessor cooling system

4.1 Database

In this project, the database is used to store all the sensor readings acquired by the Arduino UNO board. Sensor readings include the temperature of the CPU being cooled, the water flow rate of the system, the humidity level and the temperature of the water. The chosen platform to host the database on the cloud for this project is the Thingspeak platform. The Thingspeak database is updated every 45 seconds with the new sensor readings from the Arduino UNO. All the information stored will be kept for 72 hours before being erased from the database. Hence, the database will contain the last 72 hours of data which will be made available to the user either on the Android application or the Thingspeak website. Also, users can log into their accounts to obtain graphs and visualization tools to analyse the stored data in the database. Figure 4-2 shows how the data from the sensors are displayed in graphical form on the Thingspeak website.



Channel ID: 948045
Author: mwa0000017170715
Access: Private

Channel Stats
Created: 2 months ago
Last entry: 16 days ago
Entries: 647

Figure 0-28: Screen shot of the Thingspeak Database

4.2 Android application

This section details the design of the Android application and the reasoning behind the choices made for the GUI. The application was developed using Android Studio, and therefore is only available for Android devices. The application was added to the system to facilitate the users to control and monitor the cooling system from their phones. In an Android application, an activity is defined as a single screen with a GUI [13]. Aside from the login activity and the register activity, three more activities have been developed, namely, the Main, the Temperature Tracking and the More activity.

The Android application is designed for multiple users and users can login or register with a new account upon opening the app. This way, multiple users can use the same system, and it can be monitored from different devices. Figure 4-3 shows the Login activity.

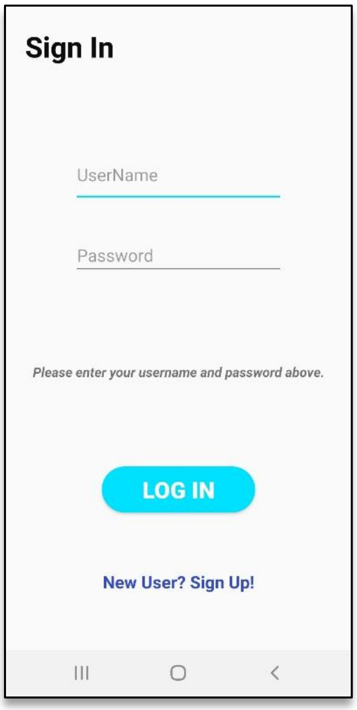


Figure 0-29: Log in Activity in Android phone

4.2.1 Main Activity

The Main activity is the first activity the user sees upon logging in, as seen in Figure 4-4. This activity is designed to control and monitor various components of the system that change over time. The application allows the user to choose between two methods of control and this can be chosen in the top right corner. The temperature controller options are the ON/OFF controller and the PID controller (refer to section 3.5 for more details).

The top grey oval button is for the user to set the desired temperature in Celsius, at which the CPU must be kept at. This temperature reading is sent to the controller Arduino via the webserver Arduino, which then controls the cooling power that is needed to cool down the load to the specified temperature. To the left side of the "Set Temperature" setting, is the power button of the system. Only when the power button is in the ON position, the user can set the temperature of the CPU is then the temperature is displayed in the blue circle located at the center once the user turns on the system.

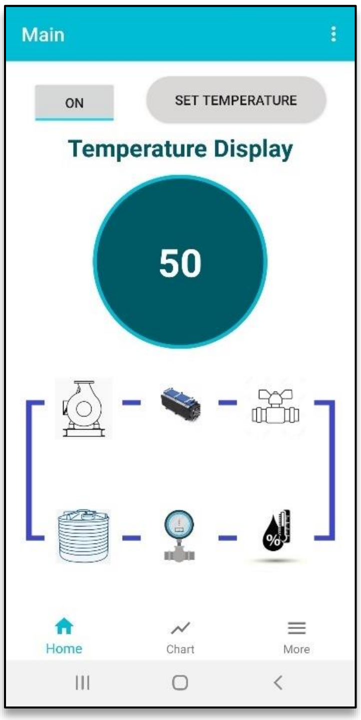
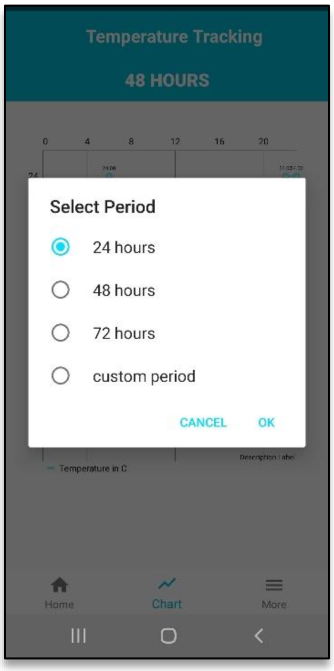


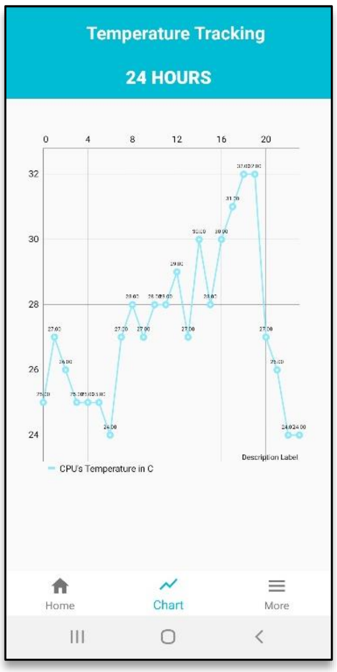
Figure 0-30: Main activity in Android phone

At the bottom of the activity, there is a simplified version of the cooling system that is represented by image buttons. The buttons show the current status of each the component of the system to the user. The use of image buttons allowed for a less clutter alternative for displaying multiple values. In clockwise direction starting from the top left is the pump, the cooling module, the solenoid valve, the humidity and temperature sensor, the water flow sensor and the water level sensor.

4.2.2 Temperature Tracking Activity

In this activity, the user can see the temperature changes of the CPU over a specified period of time. The layout of this activity was kept simple, since the majority of the screen is used by the graph, as shown in Figure 4-5(a). The time period can be chosen from the top button which displays the time period options, as shown in Figure 4-5(b). The user has three options: 24 hours, 48 hours, and 72 hours. Depending on the selected time period, the x-axis of the chart changes to accommodate the time period.





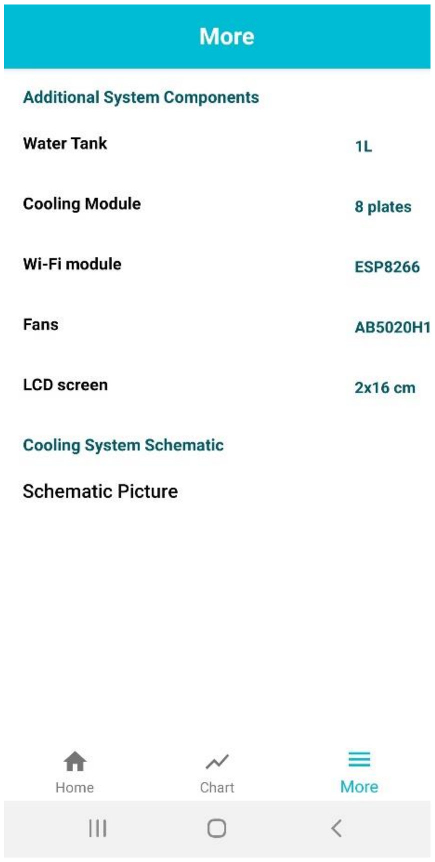
a) Layout of the Activity

b) Dialog box for time period

Figure 0-31: Temperature Tracking activity in Android phone

4.2.3 More Activity

The more activity is used to display additional information to the user about the parts of the system, as shown in Figure 4-6. In the left column is a list of components in the system that have static values. In the right column are the values or model specifications of each component. There is also a button that is used to show the user a schematic of the cooling system. This activity is generated in order to allow the user to utilize the information given, in case of a hardware malfunction.



Water Tank	1L
Cooling Module	8 plates
Wi-Fi module	ESP8266
Fans	AB5020H1
LCD screen	2x16 cm

Figure 0-32: More activity in an Android application

4.3 Arduino Webserver

Figure 4-7 illustrates the distribution of tasks between the webserver Arduino and controller Arduino to monitor all the sensors and any client connection at the same time.

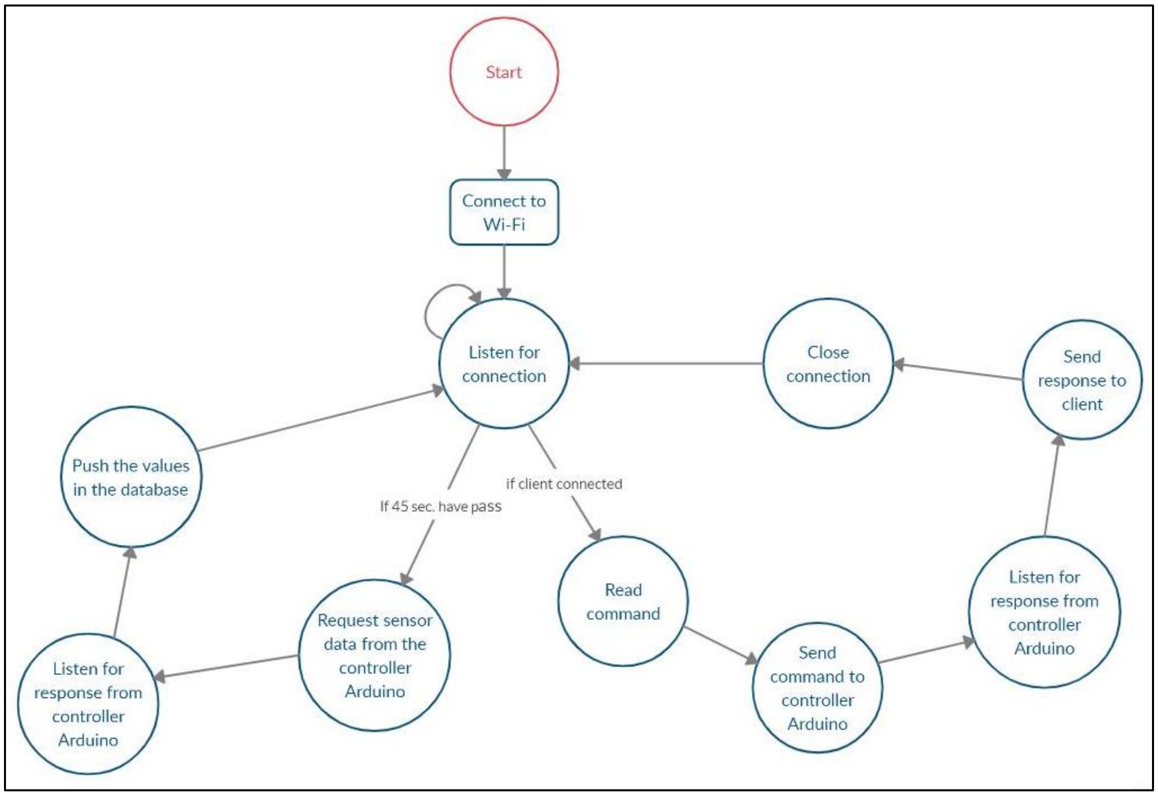

Start
Connect to Wi-Fi
Listen for connection
Close connection
Send response to client
Push the values in the database
if client connected
If 45 sec. have pass
Read command
Listen for response from controller Arduino
Request sensor data from the controller Arduino
Send command to controller Arduino
Listen for response from controller Arduino

Figure 0-33: State diagram of the webserver.

Two separate Arduinos were used as it was found that a single Arduino did not have enough memory to control the sensors as well as the webserver. Furthermore, this memory issue was causing stability problems in the system. Using two Arduino solved both the memory and stability problems. Also, since these two Arduinos can work concurrently, this makes the cooling system faster and more efficient.

The purpose of the webserver Arduino is to act as communication platform to allow communication between the controller Arduino, the database, and the Android application. Figure 4-8 describes the state flow diagram of the webserver Arduino. Upon start-up, the webserver

Arduino gets connected to the Wi-Fi network. Once connected, the webserver Arduino starts listening for a new client to connect to it. Once a connection is received from the Android application, the webserver Arduino gets the commands and forwards them to the controller Arduino. After getting a response from the controller Arduino, the webserver Arduino responds to the query of the Android application with the response obtained from the controller Arduino. As soon as the Android application receives the response from the webserver Arduino, the connection between them ends. Now, the webserver Arduino starts listening for a new connection again. Furthermore, the webserver Arduino requests all the relevant current sensor readings from the controller Arduino and pushes them to the database every 45 seconds.

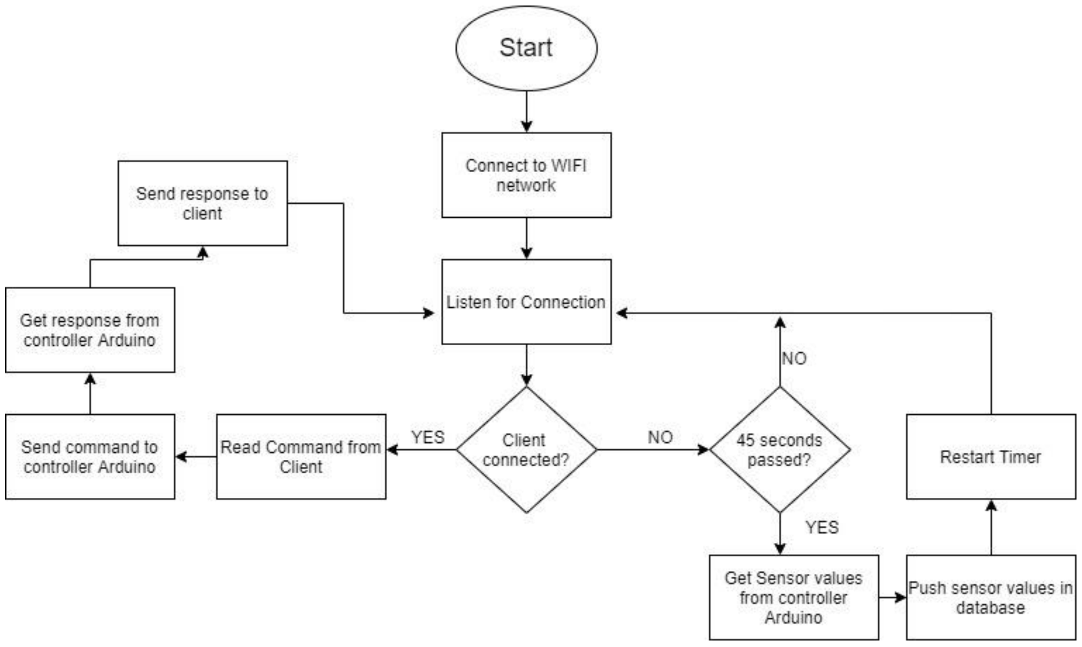


Figure 0-34: Flow diagram of webserver Arduino

5 Communication Links

The closed-loop liquid cooling system requires communication between 5 devices namely, the webserver Arduino, the controller Arduino, the Android application, the database, and the PC of which the CPU is being cooled. The chosen methods of communication and how they work are discussed in the sections to follow. Figure 0-35-1 shows the sequence diagram of all the communications links from the moment a user clicks a button in the Android application until it receives a response from the web server.

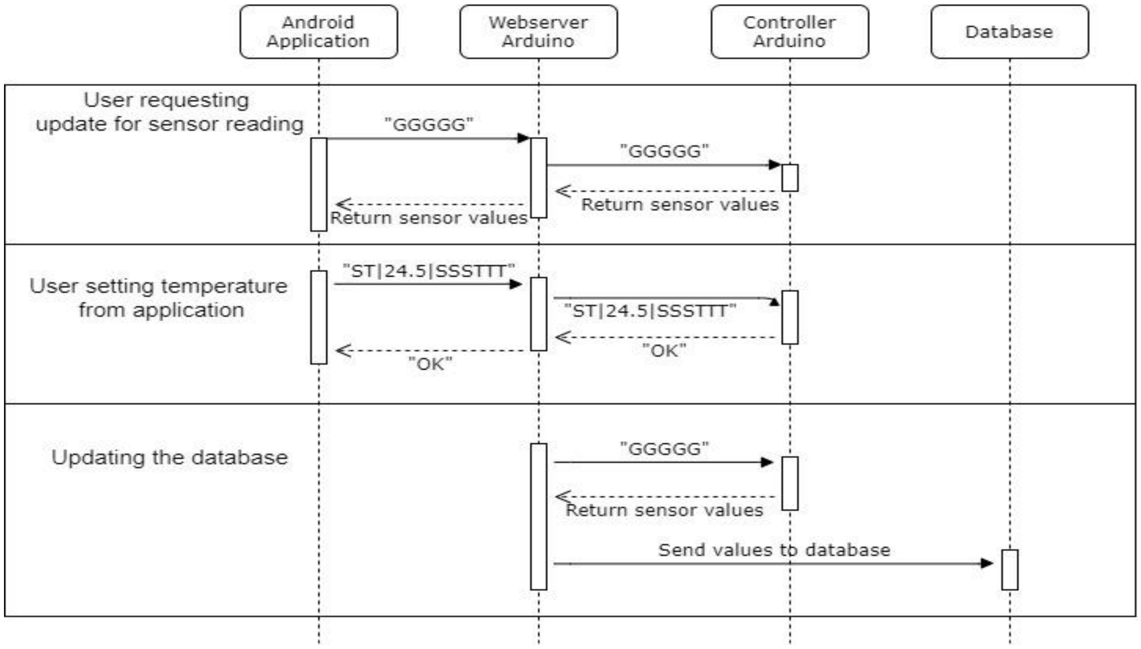


Figure 0-35: Sequence diagram

5.1 Communication between Android app and Arduino webserver

Figure 5-2 shows the class diagram for the Android application. The application has been built as a multi-threaded client. That is, once a button is clicked on the application by the user, a separate thread is created to handle the command. Thus, allowing the main thread of the application to handle any future commands from the user. This way, the application will not be lagged or delayed in case, one of the connections is taking more time than expected. Table 12-1 in the appendix describes the different commands exchanged between the Android application and the web server Arduino. When a button is clicked on the application, the userCommandHandler class will check which command should be sent and sends it to the webserver Arduino. The client class initiates a communication to the webserver Arduino via TCP/IP communication protocol by using its static IP address and port number 80. Once the webserver Arduino responds, the serverMessageHandler class handles the incoming message and provides feedback to the user about whether the command was successfully handled or not. When the Android application receives the information, the communication channel is closed by closing all the sockets and finally terminating the thread. The information received is passed to the main thread so that the GUI of the application can be updated.

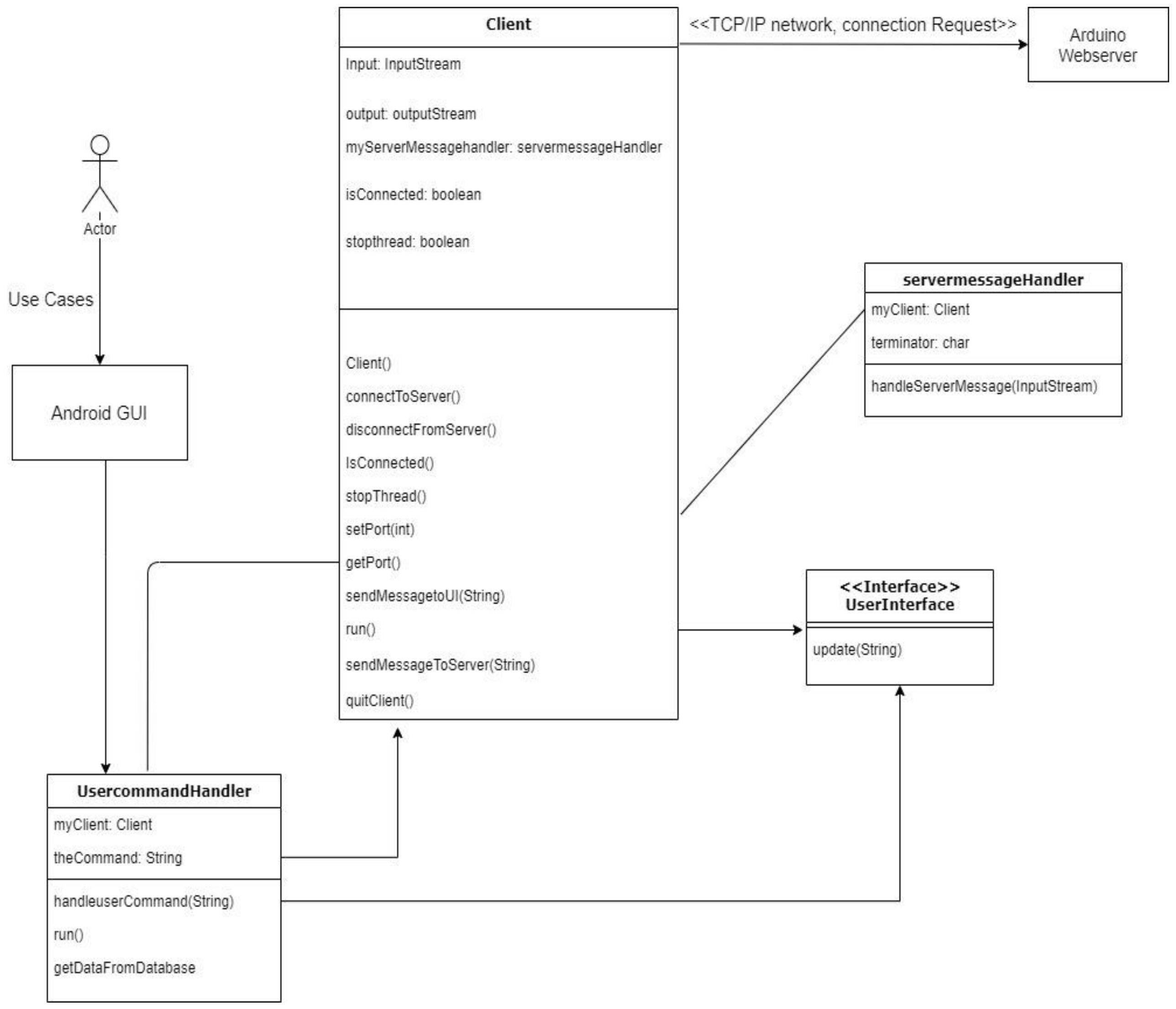


Figure 0-36: The class diagram of the Android application.

5.2 Communication between webserver and database

The webserver Arduino collects sensor readings every 45 seconds from the controller Arduino and pushes them to the database using HTTP communication. The data sent to the database are the temperature sensor of the CPU, the water flow rate inside the pipe, the humidity level and the temperature of the water flowing through the TEC elements. Since the free version of the Thingspeak platform has a limited number of times to write to the database per day (~8200 messages/day), the sensor readings are sent all at once. Also, sending data at a rate of 1 sample every 45 seconds ensures that this limit is not exceeded, while at the same time, meeting the performance metric of this project.

5.3 Communication between Android application and database

The Android application connects to the Thingspeak database to fetch sensor values that were previously stored onto the database. Each time the user selects a time period in the temperature tracking activity in the Android application, the application starts an HTTP connection with the database so that these values can be pulled and displayed to the user. The application only requests values within the time period specified by the user and are received in JSON format. Once received, the sensor values are extracted from the JSON object, plotted and displayed to the user.

5.4 Communication between webserver Arduino and controller Arduino.

The webserver Arduino and the controller Arduino communicate between each other using the I^2C protocol. According to the standard I^2C protocol, the webserver Arduino and the controller Arduino are referred to as the master and slave respectively. Pin A4 and A5 of the master Arduino are connected to Pin A4 and A5 of the slave Arduino. Pin A4 is the data line (SDA) over which data is transferred. Pin A5 is the clock line (SCL) which deals with the synchronization of the communication. The master Arduino is also connected to the ESP8266 Wi-Fi module, while the slave Arduino is connected to all the other sensors in this system. The Figure 5-3 shows the webserver and controller communication architecture.

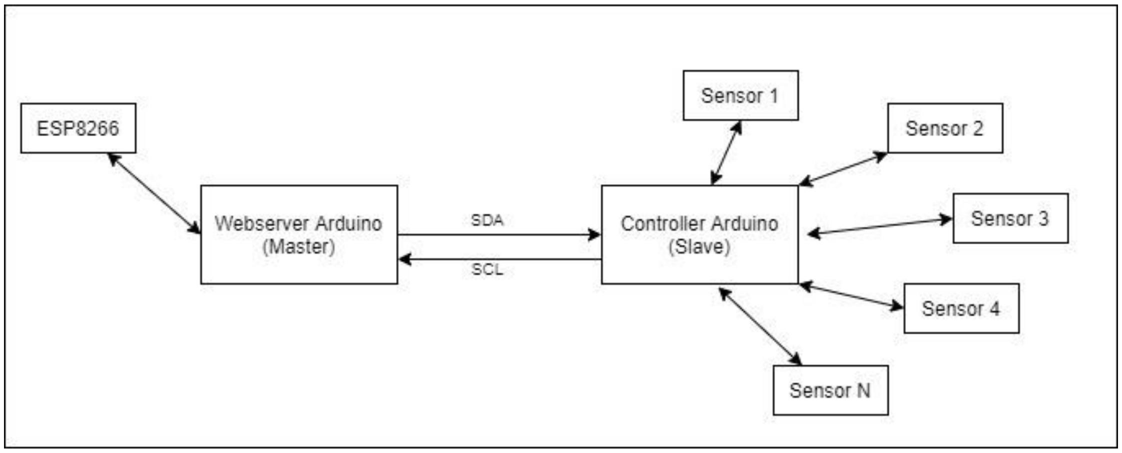


Figure 0-37: Communication architecture diagram

5.5 Communication between PC and control Arduino.

The CPUThermometer application is used to read the temperature of the CPU temperature sensors. A python script was developed to extract the CPU temperature that was read by CPUThermometer every 200 milliseconds. The rolling average is then computed with the five previous temperature readings. Thereafter, at every second, the rolling average is sent to the controller Arduino using the serial communication protocol. Using a serial port on the PC with a baud rate of 9600, the temperature is sent to the controller Arduino via USB cable. The python script is attached in the Appendix.

6 Placement Set up

Two 50 x 76 x 0.2 cm foam sheets were partitioned into sections in order to cover each hardware component in the system from the sides. Holes with a diameter of 0.9375 cm were drilled on the partitioned sheets to connect each component through the 0.9375 cm plastic tubing.

These partitioned foam sheets were then glued to the bottom of a 60 x 47.3 x 33.7 cm plastic container with water-resistant glue. Finally, holes were also cut into the sides of the plastic container to allow the heatsink and fans to extrude from the container and thereby send hot air out of the system to the ambient air. Holes were drilled into the side to pass the USB cable and the power supply cable to connect them to the PC and the 120 V outlet respectively. The final enclosure of the closed-loop liquid cooling is shown in Figure 6-1.

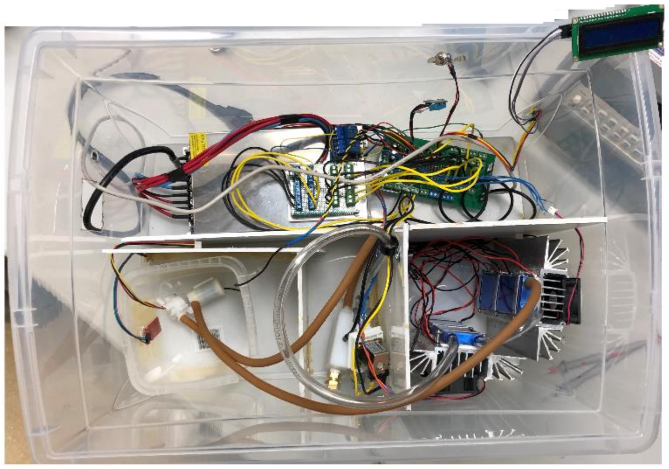


Figure 0-38: Enclosure of the cooling system

7 Project Results

- The specific heat capacity of water, the volume of 1 L of water and the change in temperature over a period of 1 minute were substituted to the fundamental formula of thermodynamics which is shown in equation 1, to determine the cooling power of the system. It was calculated to be around 253W as discussed in section 2.1.1.
- Using the On/Off controller with a static thermal load the system was able to achieve a temperature stability of ± 0.5^{o}C. Whereas when the PID controller was implemented, the temperature stability could be reduced to ± 0.3^{o}C. Temperature stability testing is discussed in section 3.5.

- The Arduino board logs the sensor data onto the Thingspeak database continuously, for more than 48 hours at a rate of 1 sample every 45 seconds.
- An alert is triggered from the controller Arduino, when the rolling average temperature of the CPU goes 5^{o}C above the set point temperature by sending a push notification to the user via IFTTT application.
- The Arduino triggers an alarm with a delay of 200 ms once the CPU's rolling average temperature goes above 5^{o}C above the user set temperature.
- After calibrating the water flow sensor, the pump was confirmed to recirculate the water at a rate of 60 L/H.
- The CPUThermometer application reads the internal CPU temperature sensor and transmits it at a rate of 1 sample per second to the controller Arduino.
- The flow rate sensor takes 60 milliseconds to read the flow rate of the system. Therefore, the Arduino has enough time to sample every minute.

8 Future Considerations

This section of the report discusses potential future considerations of the project in two parts, the first being upgrades to the existing system, and the second being additional features that could improve the system.

Currently the system uses a 3rd party application to send push notifications to the user, which means that there is more software for the user to download. To reduce the number of applications the user needs, a project improvement to allow push notifications to be sent from the system's Android application would be beneficial.

The second improvement is to create an online server having a static IP address which will continually track the changing IP address of the webserver Arduino. This will prevent the user from manually updating the IP address in the Android application code. Thus, making the device portable and user friendly.

An additional feature that could be added to the system would be to allow multiple systems to be controlled at one time. This would allow the user to cool multiple CPUs while controlling them on one Android device.

Finally, more compact and powerful cooling modules could be used. This will decrease the overall size of the cooling device. Also, if the cooling modules are more powerful, it can potentially be designed to cool multiple CPU's at the same time.

9 Conclusion

The aim of this project was to design and build a low power closed-loop liquid cooling system to cool a CPU to a temperature which is user selected through an Android application. This project was successfully accomplished while meeting all the performance metric specifications under a static thermal load. Implementing the system on a dynamic thermal load such as a PC is not yet tested due to scheduling conflicts. These testing is scheduled for the near future.

The control system of this project has been successfully implemented with two types of temperature controllers, namely, the on/off controller and the PID controller, to control the temperature of a static load and bring down its temperature to the user set temperature. Furthermore, an LCD screen was introduced into the system in order to monitor the temperature of the CPU when the user is not using the Android application.

The PCB was designed in order to connect the electrical circuit components of the system in a more organized manner to readily troubleshoot if there are any problems in the electrical circuit. Moreover, the trace thickness was carefully chosen to design the PCB, so that it can withstand the high current that is consumed by the TEC plates.

An emergency shutoff valve system was also implemented to ensure the safety of the system. When a leak is detected by the water level sensor which is submerged inside the water tank, it activates the solenoid valve electromechanically thereby shutting off the system. This mitigates the risk of damage caused to the system due to the contact of water with electrical components. An alert notification setting has also been developed via IFTTT application to notify the user if there is any leak in the system. Furthermore, a similar alert is also sent to the user when the rolling average temperature of the CPU goes 5 °C above the set temperature.

In order to allow the user to control and monitor the cooling system remotely, the Android application was designed using Android Studio. This Android application allows the user to choose between the on/off controller and the PID controller to cool the CPU, depending on the needed precision of control. The user can also set the temperature at which he/she wants the CPU to be cooled and maintained. In addition, the user also to monitor the CPU temperature as well as the status of the system devices which includes the pump, the TEC module, the electric valve, the humidity and temperature sensor, the water flow sensor, and the water level sensor. The Android application is also capable of displaying the temperature trend of the CPU over a time period of either 24 hours, 48 hours or 72 hours, as selected by the user.

Though the main goal of this project was to design and build a closed-loop liquid cooling system to cool a CPU, the system has only been tested on a static thermal load. The results concluded the system can cool a static load, initially at room temperature, to 17 ± 0.5^{o}C when an on/off controller is implemented and 17 ± 0.3^{o}C with a tuned PID controller. Therefore, this system can be utilized in conventional water cooling in fish tanks and high-purity water cooling in manufacturing plants [2]. In addition, by upgrading this system to a very high-performance system with a higher cooling temperature range, it would be possible to integrate this device for liquid cooling purposes in the information technology, medical, transportation and military fields.

10 References

[1] P. Bates, “How Heat Affects Your Computer, And Should You Be Worried?” MakeUseOf, 27-Aug-2014. [Online]. Available: https://www.makeuseof.com/tag/how-heat-affects-your-computer-and-should-you-be-worried/. [Accessed: 06-Feb-2020].

[2] “Computer cooling,” Wikipedia. [Online]. Available: https://en.m.wikipedia.org/wiki/Computer_cooling#Liquid_cooling. [Accessed: 18-Feb-2020].

[3] I. Cutress, “The AnandTech Coffee Lake Review: Initial Numbers on the Core i7-8700K and Core i5-8400,” RSS, 05-Oct-2017. [Online]. Available: https://www.anandtech.com/show/11859/the-anandtech-coffee-lake-review-8700k-and-8400-initial-numbers/5. [Accessed: 19-Feb-2020].

[4] “Thermoelectric cooling,” Wikipedia, 29-Jan-2020. [Online]. Available: https://en.wikipedia.org/wiki/Thermoelectric_cooling. [Accessed: 18-Feb-2020].

[5] “Tec1 12706 12v 6a Thermoelectric Cooler Peltier Module,” *indiamart.com*. [Online]. Available: https://www.indiamart.com/proddetail/tec1-12706-12v-6a-thermoelectric-cooler-peltier-module-20744920097.html. [Accessed: 07-Mar-2020].

[6] “Thermodynamic equations,” *Wikipedia*, 16-Nov-2019. [Online]. Available: https://en.wikipedia.org/wiki/Thermodynamic_equations. [Accessed: 07-Mar-2020].

[7] “DIY Thermoelectric Cooler Refrigeration Air Cooling Device 8-Chip TEC1-12706 12V 576W,” *Amazon.ca: Home & Kitchen*. [Online]. Available: https://www.amazon.ca/Conditioner-8-Chip-TEC1-12706-Thermoelectric-Cooling/dp/B073RCY438. [Accessed: 07-Mar-2020].

[8] Thermonamic, “Specification of Thermoelectric Module TEC1-12706,” High Performance

and Highly Reliable Solution for Cooling and Heating Applications, pp. 1–3.

[9] “ThingSpeak,” Wikipedia, 07-Feb-2020. [Online]. Available: https://en.wikipedia.org/wiki/ThingSpeak. [Accessed: 29-Jan-2020].

[10] “Advanced Circuits,” *Printed Circuit Board Trace Width Tool | Advanced Circuits*.[Online]. Available: https://www.4pcb.com/trace-width-calculator.html. [Accessed: 07-Jan-2020].

[11] “PID Temperature Controllers,” West Control Solutions. [Online]. Available: https://www.west-cs.com/products/l2/pid-temperature-controller/. [Accessed: 19-Feb-2020].

[12] “PID Control made easy,” Eurotherm by Schneider Electric, 04-Dec-2018. [Online]. Available: https://www.eurotherm.com/temperature-control-us/pid-control-made-easy/. [Accessed: 18-Jan-2020].

[13] “Android - Activities,” Tutorialspoint. [Online]. Available: https://www.tutorialspoint.com/android/android_acitivities.htm. [Accessed: 24-Jan-2020].

12 Appendix

Table 12-1. Commands Table

Commands	Description	Example	Response Example
“GGGGG”	Requests temperature, flow rate and humidity sensor readings.	“GGGGG”	TTTT|FFFF|HHHH E.g 45.59|60.45|10.75
“ST|**XXXXX**|SSSTTT”	Set the desired temperature where **XXXXX** is the desired temperature to 2 decimal places.	“ST|26.45|SSSTTT”	“OK”
“PPPPP”	Toggle the status of the pump. If the pump is on, turn it off and vice versa.	“PPPPP”	“OK”
“VVVVV”	Toggle the status of the valve. If it was open, close it and vice versa.	“VVVVV”	“OK”
CCC|OOO|CCC	The on/off controller is selected.	“CCC|OOO|CCC”	“OK”
CCC|III|**XX**|**YY**|**ZZ**|CCC	The PID controller is selected. The value of **XX**, **YY** and **ZZ** are the K_p, K_i and K_d respectively.	“CCC|III|135|245|756|CCCC”	“OK”

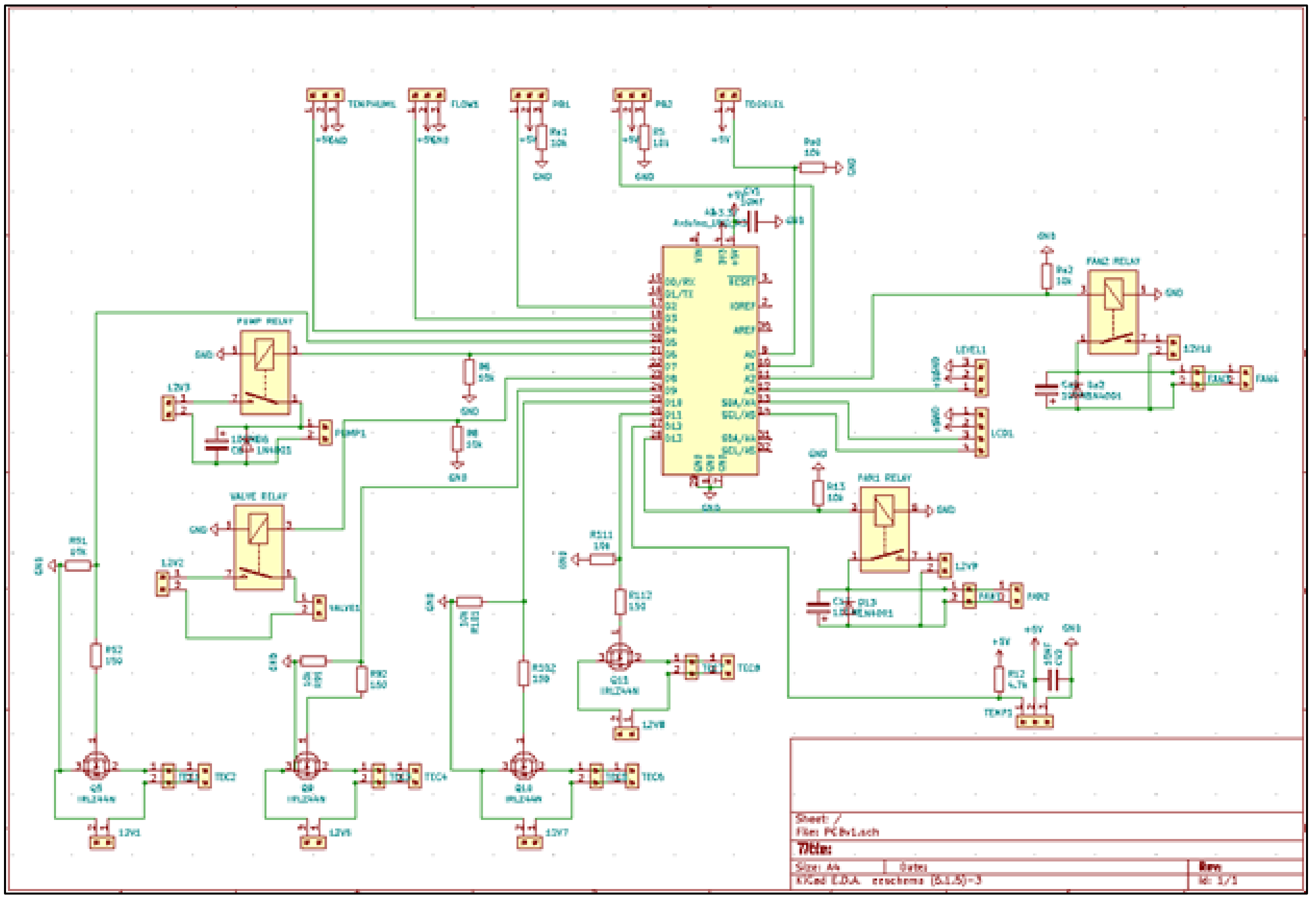


Figure 0-39: Schematic of the control system