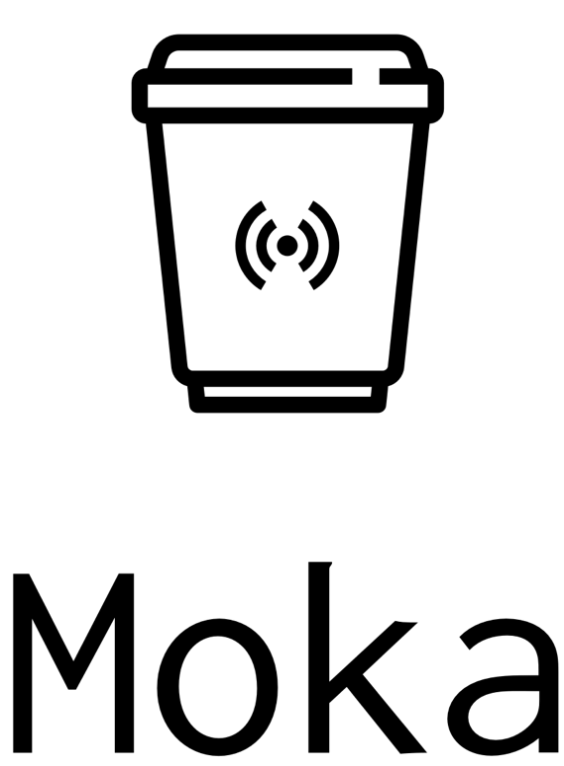
Moka - RFID Custom Coffee Machine



College of Engineering and Computer Science

University of Central Florida

Advisors: Samuel Richie, Lei Wei

Senior Design 2 Final Documentation

Summer 2018

Group 18 Members

Daniel Betancourt - Electrical Engineering

Guilherme Carvalho - Electrical Engineering

Dylan Kirke - Computer Engineering

Alessandro Vecchi - Computer Engineering

**Contents**

**List of Figures**

**List of Tables**

**1 Executive Summary** 1

**2 Product Description** 2

2.1 Motivation 2  
2.2 Goals and Objectives 3  
2.3 Requirements and Specifications 4  
2.4 UI Examples/User Interaction 5  
2.5 House of Quality 7

**3 Design Constraints and Standards** 9

3.1 Standards and Other Safety Concerns 9

3.1.1 Soldering Standards 9  
3.1.2 RoHS Compliance 10

3.1.3 SCAA Water, Roasting, and Brewing Standards 10

3.1.4 ASTM Heated Surfaces Safety (ASTM code C1055-03) 11

3.1.5 C Standards 12

3.1.6 JavaScript Language Standards 13

3.1.7 Wi-Fi Standards 14

3.1.8 RFID Standards 14

3.1.9 Milk Storage Standards 15

3.1.10 Git Standards 16

3.2 Project Constraints 17

**4 Research and Background Information** 19

4.1 Microcontroller Considerations 19

4.1.1 Particle Photon(P0 Module) 19

4.1.2 ESP32 20

4.1.3 ESP8266/ATmega328P 21

4.1.4 Arduino MKR1000(SAMD21 Cortex-M0+) 22

4.2 Serial Communication Technologies 23

4.3 Wireless Communication Technologies 24

4.3.1 RFID 24

4.3.2 Bluetooth 26

4.3.3 Wi-Fi 26

4.4 App Development 27

4.4.1 Frontend 27

4.4.2 Backend/Database 30

4.5 Power Technologies 32

4.5.1 Power Supplies 33

4.5.2 Power Distribution Board 34

4.5.3 Solid State Relays 34

4.6 Software Tools 36

4.6.1 Communication 36

4.6.2 Development 38

4.6.3 Documentation 39

4.7 PID Controller 39

4.7.1 Why PID 42

4.7.2 Parallel Op Amp PID Controller 43

4.7.3 Series Op Amp PID Controller 45

4.7.4 Difference Amplifier 45

4.7.5 Op-amp Astable Multivibrator 46

4.7.6 Power Amplifier 47

4.7.7 Temperature Sensor 48

4.7.8 Op-amp Parameters 49

4.7.9 Op-amp Comparison 50

4.7.10 Tuning Procedure for PID Loop 52

4.7.11 Troubleshooting Common PID Problems 53

4.8 Type of Boilers 53

4.9.1 Heating Elements 53

4.9 Refrigeration Unit 55

**5 Design** 57

5.1 Hardware Design 57

5.1.1 Hardware Block Diagram 57

5.1.2 Microcontroller 58

5.1.3 RFID Module 60

5.1.4 PID Controller 62

5.1.5 PID Box 65

5.1.6 Coffee Dispenser 66

5.1.7 Boiler70

5.1.8 Water Tank 71

5.1.9 Refrigerator 72

5.1.10 Design casing 75

5.1.11 Power Module 76

5.1.12 Peristaltic Pump 79

5.1.13 Filter Cleaning 81

5.2 Software Design 83

5.2.1 Software Block Diagram 83

5.2.2 Database 84

5.2.3 App Backend Development 89

5.2.4 App Frontend Development 92

5.2.5 User Interface/User Experience 94

**6 Prototyping** 101

6.1 Printed Circuit Board 101

6.1.1 PCB Practices 101

6.1.2 PCB Design 102

6.1.3 PCB Vendors 106

6.2 Prototype Expectations 107

6.2.1 Potential Hardware Issues 107

6.2.2 Potential Software Issues 108

6.2.3 Prototype Constraints 109

6.3 Parts Acquisition and Bill of Materials 110

**7 Testing** 112

7.1 Hardware Testing 112

7.1.1 Microcontroller Testing 112

7.1.2 RFID Scanner Testing 112

7.1.3 PID Controller Testing 115

7.1.4 Power Supply Testing 116

7.1.5 Boiler Testing 117

7.2 Software Testing 118

7.2.1 Microcontroller Software Testing 119

7.2.2 Database/Server Testing 121

7.2.3 RFID Testing 124

**8 Administrative Content** 126

8.1 Division of Labor 126

8.2 Project Milestones 126

8.3 Budget and Finance 127

**9 Conclusion** 129

**Appendix A References** 130

**Appendix B Permissions** 133

**List of Figures**

1 - Coffee station 5

2 - Delete cup 5

3 - Delete card 5

4 - App menu 5

5 - Cups page 5

6 - Scan cup 5

7 - Flavor select 6

8 - Cream select 6

9 - Sugar select 6

10 - Payment info 6

11 - Add card 6

12 - Cup size 6

13 - House of Quality 7

14 - Temperature-Time Relationships for Burns 11

15 - Example of a Power Distribution Board 30

16 - Solid State Relay Circuit Schematic 31

17 - Illustration of the zero point crossover of a SSR 31

18 - Closed Loop Diagram of PID 35

19 - PID closed loop diagram with gain terms 36

20 - Proportional Band Explanation 36

21 - Correctly Configured Proportional-Band 37

22 - Average extraction yield relative to brewing temperature in Espresso 38

23 - Analog Implementation of a PID controller using operational amplifiers 39

24 - Series PID diagram (top) and op-amp implementation (bottom) 40

25 - Difference Amplifier for op-amp PID 41

26 - Astable Multivibrator using Op-amp 42

27 - Op-amp pinout (left) 46

28 - Thermoelectric Cooling module 50

29 - Hardware Block Diagram 51

30 - Arduino MKR1000 Schematic (1/2) 52

31 - Arduino MKR1000 Schematic (2/2) 53

32 - Arduino MKR1000 Board Layout 53

33 - MFRC522 Block Diagram 54

34 - ID-12LA and ID-20LA Pinout 55

35 - Op-amp PID schematic with difference amplifier input 56

36 - Proportional Term Simulation 57

37 - Integral Term Simulation 57

38 - Derivative Term Simulation 57

39 - Circuit used to provide power to op-amps 58

40. Op-amp multivibrator used to make output into a PWM signal 58

41 - Wiring of Solid state Relay with dinkle controller inside PID box 59

42 - Ground Dispenser 60

43 - Stepper Motor Dimensions 61

44 - Formula used for Current Limiting 62

45 - Dispenser Subsystem Schematic. 63

46 - Wiring of Solid State Relay to control boiler 64

47 -Electronic Semiconductor Refrigerator Radiator Cooler 67

48 - HiLetgo Digital Temperature Controller Board 68

49 - Model coffee machine 69

50: Final Implementation of Case 69

51 - SP150-12 71

52 - MK40S-5 72

53 - Small Peristaltic Pump (left) and Larger Peristaltic Pump (right) 74

54 - HIC Plastic Coffee Filter 75

55 - SunFounder Metal Gear Digital RC Servo Motor 76

56 - Software Block Diagram 77

57 - Database model diagram 81

58 - Data Flow Model 84

59 - React Redux Data Flow 85

60 - Application Login 88

61 - Create Account 88

62 - Coffee station page 89

63 - App navigation menu 89

64. Flavor select 90

65. Cream select 90

66. Sugar select 90

67 - Cups page 91

68 - Scan page 91

69 - Size select 91

70 - Payment info page 92

71 - Add payment info 92

72 - Delete coffee 93

73 - Delete cup 93

74 - Delete card 93

75 - Preliminary PID Circuit Design 95

76 - PID Design Schematic 98

77 - PID PCB Design 99

78 - Final PID Schematic Design 99

79 - Final PID PCB Design 100

80 - Final Microcontroller PCB Design 100

81 – Final Built PCB Design 101

82 - Program Stack Illustration 104

83 - 24-pin ATX connector 110

**List of Tables**

Table 1 - Water Quality Specifications 10

Table 2 - Wi-Fi Standards 13

Table 3 - RFID Standards 14

Table 4 - Particle P0 Module Basic Specifications 17

Table 5 - ESP32 Basic Specifications 18

Table 6 - ESP8266 Basic Specifications 19

Table 7 - ATmega328P Basic Specifications 20

Table 8 - SAMD21 20

Table 9 - Basic RFID Frequency Info 22

Table 10 - Development Tools 34

Table 11 - Controller Transfer Functions 40

Table 12 - Sensor Type Comparison 43

Table 13 - Op-amp Parameters and their definitions 43

Table 14 - Low-Noise Op-amp Comparison 46

Table 15 - Moka RFID pin connections 55

Table 16 - A4988 Resolution Chart 61

Table 17 - Technical Specifications of SP150-12 71

Table 18 - Technical Specifications of MK40S-5 72

Table 19 - Technical Specifications of small and large peristaltic pumps 74

Table 20 - Technical Specifications of the SunFounder Servo Motor 76

Table 21 - User Schema 79

Table 22 - Machine Schema 80

Table 23 - User Model Specifications 80

Table 24 - Machine Model Specifications 81

Table 25 - Authentication Routes 83

Table 26 - User Routes 83

Table 27 - Coffee Routes 84

Table 28 - Actions and Reducers 87

Table 29 - List of all components required for PID PCB design 97

Table 30 - parts list 105

Table 31 - Division of labor 118

**1 Executive Summary**

The project is to design a coffee vending machine that is able to brew and serve coffee to users. The coffee machine is able to detect which user is accessing the machine using an RFID sensor. The user will have an RFID sticker attached to their cup. This sticker will contain a unique ID for that specific cup which was attached to the user’s account previous to accessing the machine. The microcontroller utilizes the WINC1500 low power 2.4GHz Wi-Fi module to make a GET request to the backend of our application using the unique ID retrieved by the RFID module. The server then responds with the user’s default coffee preset settings and the machine begins brewing the coffee. The coffee preferences are all defined by the user on the Mocha mobile application. On the application, users are able to register cups, modify coffee setting presets, and modify payment information.

The machine is designed to be used in a busy, commercial environment where it can be used frequently by many people. The idea is for people to have a convenient way to get their coffee without having to configure any settings on the machine or having to provide a payment method on the spot. The machine is able to brew the coffee within one minute of scanning the coffee cup furthering the machines viability in a commercial environment.

The device is powered via 120V AC from a wall outlet and then be scaled down to 5V DC to power the microcontroller and RFID sensor. The power supply is to be stored in the lower end of the coffee machine.

To configure the SSID and password, we will be uploading a firmware specific to each individual coffee machine. This firmware contains the SSID and the password to the locations Wi-Fi access point.

The total size of the machine is 30 inches in height, 20 inches in width, and 25 inches in depth. This gives us enough space to house the electronics, microcontroller, PCB, boiler, and PID controller.

**2 Product Description**

The following section gives a basic description of Moka’s system design. This section covers the team’s motivation as well as goals and objectives behind the overall system design. This section also covers the requirement specifications the design has to meet, a quick example of our application UI, and a House of Quality diagram that gives the team a good idea on the balance of requirements that makes our product more desirable to the market.

**2.1 Motivation**

The driving force of a good product is the ability for it to solve a need. We were motivated to create this product because observed saw the desire people had for a simple and fast way to obtain their everyday cup of coffee. We observed the long lines that people had to wait in at the on-campus coffee shops to simply obtain one cup of coffee. There was a need for a fast-premium cup of coffee.

To come up with the design we first had to ask ourselves a few questions. Why do people currently avoid using coffee vending machines to get coffee fast? How can we entice people to keep coming to our machine instead of going to a coffee shop? The answer that we came up with is a fast coffee vending machine that brews coffee on demand instead of storing pre-brewed coffee. The machine also provides a free membership service powered by RFID stickers that the users attach to their favorite cup. Whenever a person with an RFID cup sees any of our coffee vending machines, all they have to do is place their cup on the scanner and the machine looks up their desired coffee and payment information along with any coffee specifics such as cream amount and sugar amount from our database and begin brewing the coffee of their choice.

The most efficient way we thought of to communicate with the machine was through a mobile application. In the application, the user would first register their RFID sticker using a QR code. Then they would enter their payment information and select their desired coffee grinds, cream amount, and sugar amount. They would then select their preferred coffee settings for their next visit to one of the machines. We were motivated to create a mobile application because it gave users the ability to change their coffee settings on the go without having to go to one of the machines to modify their coffee settings. This would also keep lines at the machines shorter without sacrificing functionality.

There are many features that we were motivated to implement but in order to save time for the users we decided not to implement. One feature we decided to avoid implementing was grinding the coffee beans on demand as well as brewing the coffee. Grinding the coffee beans would create problems like forcing the user to wait longer for each cup of coffee and would also require for maintenance personnel to come every few days to clean out the grinder to avoid stale and poorly ground coffee.

**2.2 Goals and Objectives**

The objective of this project is to design, build, test and implement a premium, and truly automatic coffee machine for commercial use. Our mission is to create an easy-to-use automatic coffee machine in order to speed up waiting times at places that sell coffee in addition to coffee. Another one of our goals is to provide advanced options that more experienced coffee drinkers can take advantage of. There are two key features that differentiate this machine from other automatic coffee machines. The first one is the use of RFID technology to facilitate the day to day transactions of our users. The second one is the quality of coffee that the machine produces at a reasonable price and a small waiting time.

There are a few problems current commercial coffee machines suffer from that have decreased their popularity amongst regular coffee drinkers. The first problem is that they are perceived as the cheap alternative to getting coffee from a coffee store. The user is not shown the different steps that the coffee machine goes through from the moment the user makes a selection to when the coffee is dispensed. This creates the illusion that the coffee is not going through the same process that a barista at a coffee shop goes through to create the same cup of coffee. Another problem is that there is no level of customization for the user. At a coffee shop, a person can ask for no sugar or light on milk but at a commercial coffee machine, the machine chooses all those things for you. One more problem is that waiting times at a coffee shop are long due to interactions with a cashier as well as by people who ask for food or more complex drinks rather than a cup of coffee.

When a user visits our coffee machine for the first time, they are asked if they have already created account. Their account stores all the pertinent information needed for any future transaction, this includes payment information, type of coffee, and all the brew settings. Once their account is created, the machine dispenses an RFID sticker that they can stick on the bottom of their favorite coffee mug or thermos. The next time the user visits the coffee machine, all he/she has to do is place their coffee mug on the dispenser and select one of their preset coffee settings and the machine begins the brewing process. Competing automatic coffee machines are not truly automatic. The only thing they automate is the brewing process. Our machine automates the user input, brewing, payment processes and provides more customization options for each user as well.

The process for creating a truly premium cup of coffee requires very specific measurements and timing. Ideally, our machine is equipped with the ability to select different grind sizes, choose optimal brew temperature, and set custom brew times. There is an option while creating a preset coffee setting where the user can manipulate these advanced settings to get the coffee they want. Users are not required to know anything about the details of coffee brewing but if they do, they have a lot of customization options that they can utilize.

Installation of the machine should be simple and require less than an hour to complete. Once the machine is delivered, the machine just needs to be plugged in to start up. Then, the client fills it up with water and whatever selection of beans they wish to have served. After that, the administrator connects it to a nearby wireless network. Once that is done the machine is ready to use.

**2.3 Requirements and Specifications**

* Physical Device
  + The device will be no larger than 30 in tall by 25 in wide by 25 in long.
  + The machine will cost less than $1000.
  + The machine must plug into a standard 120 VAC outlet.
  + The machine will deliver the coffee in less than four minutes.
  + The machine’s install time should be less than 10 minutes.
  + Water temperature in the boiler must reach 197.6°F within one minute, and never exceed 204.8°F.
  + The RFID sticker should operate at 13.56MHz frequency.
  + The machine will dispense 55g of coffee per liter of water.
  + The milk stored in the machine will be stored between 35°F and 40°F.
  + The peristaltic pump shall dispense water at 100ml/sec
* Mobile Application
  + A user will not be able to set no more than 10 different coffee preferences in their profile.
  + A user will not be able to set no more than 10 different cup RFID stickers in their profile.
  + The mobile application will be compatible with iOS and Android devices.
* Cloud Server
  + Each user instance will not have more than 10 different coffee preferences stored in the database.
  + Each user instance will not have more than 10 different cup IDs stored in the database.
  + The database will have 10 GB of storage space.
  + The connection network speed between our microcontroller and our database should be at least 15Mb/s.

**2.4 UI Examples/User Interaction**

Figures 1 through 12 showcase a brief overview of how our mobile application will look like. We will be going into more detail about each page on section 5.2.5.

|  |  |  |
| --- | --- | --- |
| https://lh6.googleusercontent.com/Te_Gc5drZkLBGrR30Vx8H0DrSKi9d58TUY6Qi2znpXrm0fymLmf-_CBvpishIkaw4WDYd_yXmRFI-kgfSEifLi6iTiNvD0u6wCH8Z7_SMN-aGEcUtjxYAQf-R3wlMCeiEVrxIfrx | https://lh4.googleusercontent.com/gEIS8OUEPBAIAKoObzO4XR77PPNAxJU4cop2aB-ePwGMl-DIhSLCXeYVXLeqwpnPNNdO2d3gWE-51jx-x1QKuPCcPl5sj_AEhzYuxnzBdcUcDCR1r9ybc49Xg2geBXp3fEOiQnvr | https://lh3.googleusercontent.com/FLwQP3gTlM5OK32Zg8ZNnorIZKpf8bLmTfEiZWMNn7M_rLEioArZiA8LIZMZRxxK2QGMKVKugMtmVuHxRfC-eldAF3M0dSCOp5aNUkv3NSYZ6-pTQow7-HAQ8IX8ZUqy4PjjUYal |
| *Figure 1 -* Coffee station | *Figure 2 - Delete cup* | *Figure 3 -* Delete card |

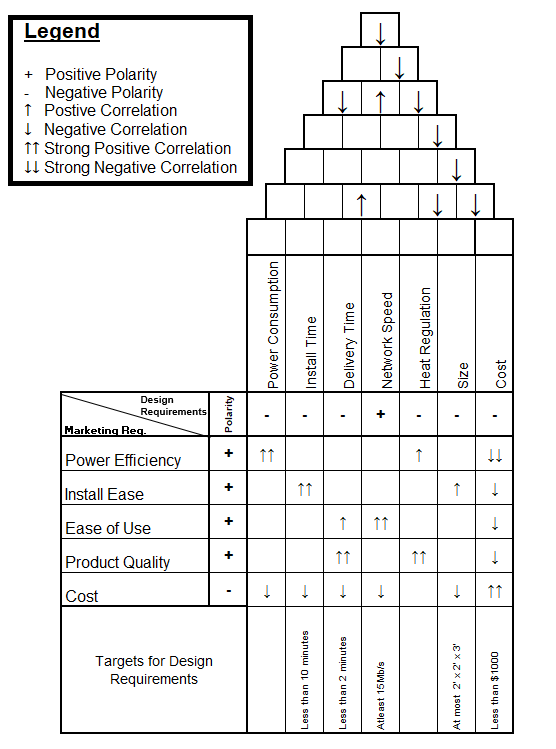
|  |  |  |
| --- | --- | --- |
| https://lh5.googleusercontent.com/kwp2I46QHxXWZEvp7j5f3gikdCOPKoSG5NkUQWGHsx05SshqNgwELGIutulBH4qsiYkwzci0m0Fw5cyFMNGdZjPsutD05MdGwiL0bhrv4BNYZT8E7rXbmmOV6-1aM8r4AKdNGBco | https://lh4.googleusercontent.com/0kMBLNm4_lQzoc0C0cUrN84XxFrtzwftNHlwy3Uz-eVMRf2_ZRH2EZBMt-zQuOwi-_LgiH4zKwxxP6r40LQFIJrn5E_8Ig5qG8FlOuiWEGaZBh1I_fMr2yDKePOnXJEYhGzbeeW_ | https://lh6.googleusercontent.com/VrzlXPlEQ48Fg4Vv5rkFadOdu5n3f874Isp4Iku_zvwztepxFGAXwvow2JN6dqP00sAazht-2VRWgNLkYl8BBSn6CpxdXcxTctvRNQHX6NDOHHND-UBU9bEeDMmjmc8cfGkIBKRO |
| *Figure 4 -* App menu | *Figure 5 - Cups page* | *Figure 6 -* Scan cup |

|  |  |  |
| --- | --- | --- |
| https://lh6.googleusercontent.com/0FJcpf87cv4CuUqLK0cVAIZR2AA_M7dhCPSoryg724Az4i7NjD1g8Z2Y2BWRLE1y_0wj5l1RWU36L0f9FoJvHL2WEbtVvT2BEguHNm5QqpdNUdXIU2XacIIlV1LxUQbN4BehE_nk | https://lh5.googleusercontent.com/w4Nh6_vdiOOR6EmhC9dVQ_M0-NIzPzhUigiDXawIVQO3GvUowh1cYTSBjdc10ew6q74Iqbq5xv6u9ERAcvo_RXPj3BQU8JVSQszd1QqnVT7oy2wrvn4aP_eyekttRvO9miFx9eta | https://lh3.googleusercontent.com/vWIfLRy58GV_GBRro3AUK135jyU8DYHPilYZbEMD4QQZc4eLOH-74pmMADJp4xgXeP1h98xDg5Zz3MRnO7Zf3jved3CEY5Ntwl3klxhJst69Y6YAGam7VzBslWnG_omxoKIGoPTF |
| *Figure 7 -* Flavor select | *Figure 8 -* Cream select | *Figure 9 -* Sugar select |

|  |  |  |
| --- | --- | --- |
| https://lh6.googleusercontent.com/ZmiIY8RlzKF4xMch8pCZiLiDrlojpDG_0AkC53Yhfu0FauxVMtGoBomSKpONSOK7beMkHLSr3Z-t724dG9PFyn9H3cQ2V8KMLENdhTcGKiNb9NmvRXKMPKIMbAhQLd3AfZkxo_AS | https://lh4.googleusercontent.com/ntnyJS-3GcftJazkslc3U1kgspwpHrrjO0BruP-isDTWwuH1HT7E9eskAX5eu3JKeW-1vvtAoaVEImlyAfrzV0IOIVmNMIDYrkD4RE8hTI7YDRRIWNScIpx_b0m3kNc_yFQg8bGq | https://lh4.googleusercontent.com/l5e3eJdWwo2RWcL5NI-_mMC6-FJ-YTjalsPEYrz6yy5rZliBEJf7jNko-WjxRk5HdjFAPoVe87DAdYfvv54pOpFhgXEeOBtbSPqqQyXCY4467FDsZqXkM2HStw6Dys2HvwQxENsk |
| *Figure 10 -* Payment info | *Figure 11 -* Add card | *Figure 12 -* Cup size |

**2.5 House of Quality**

As shown in our House of Quality diagram in figure 13, one of the main focuses of our design’s requirements is to keep the use of our system simple and convenient for our customer and users. We had to develop the perfect balance between making our system easy to setup and use while still being able deliver a high-quality cup of coffee within a reasonable time frame. If either one of these factors are more highly favored over the other, it could lead to our system being undesirable to customers. The one requirement that we must pay special consideration to would be the cost. This is because all of the other requirements are directly dependent on the cost of the system. Almost every requirement has at least one negative correlation with the system’s cost; showing that as our system gets better, the price will exponentially increase. If the system is too costly, then it won’t sell no matter what benefits it will bring to the consumer. Our design takes these factors into consideration in order to make sure that it can be made in the most cost-effective manner.



*Figure 13 -* House of Quality

**3 Design Constraints and Standards**

The following section will go into the various standards that our design with have to adhere to as well as the requirements and constraints of our system design. The standards and constraints identified in this section all pertain to our project and should be followed in order to ensure our design is successful.

**3.1 Standards and Other Safety Concerns**

This section will detail various mechanical, electrical, and software standards. Alongside each of them, there will be a discussion on how each pertains to our design. A discussion on general safety concerns and procedures in order to ensure the safety of group members throughout the project will also be found in this section.

**3.1.1 Soldering Standards**

NASA’s standards document NASA-STD-8739.3 titled “Soldered Electrical Connections” will be referred to for or any soldering that will need to be conducted on any components in the system’s design. These guidelines will be used as a baseline for what will be expected for any soldering done on the system’s PCB or other subsystems. For all the standards in this documentation, section 11 will be referred to the most because it provides detailed standards on what should be done for hand soldering of printed circuit boards. This includes standards on the preparation of the PCB. Including to make sure the PCB is clean of any debris that may have gathered on it. As well as to have the PCB demoisturized within eight hours of the boards initial exposure to soldering temperature. The soldering iron tip should be both clean and tinned. Tinning is done by adding a bit of solder to the tip, then wiping and rotating it on a damp sponge to reveal a shiny thin layer of solder. This prevents your tip from oxidizing increasing the iron’s life and allows solder to flow onto components more quickly rather than sticking to the tip. After the preparation is complete proper application must be done to ensure the quality of the deliverable system. When hand soldering onto the PCB, contact with both the PCB’s copper layer and the component with your iron must be ensured before applying solder. When applying solder to the terminals, proper coverage of the solder must ensure that “the molten solder shall flow around the conductor and over the termination areas.”(CITE) Once the soldering is complete the components will then need to be cooled at room temperature.

**3.1.2 RoHS Compliance**

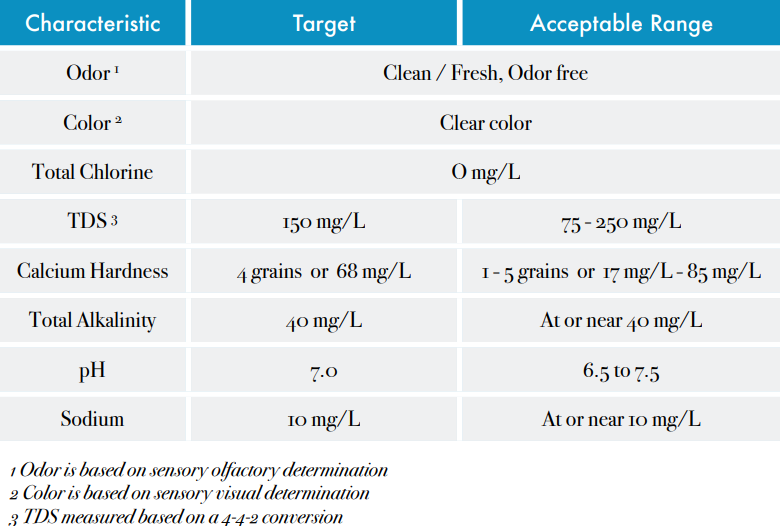
RoHS stands for the Restriction of Hazardous Substances Directive. It is shorthand for the Directive on the restriction of the use of certain hazardous substances in electrical and electronic equipment, and was signed and adopted February 2003 by the European Union. Although commonly referred as the “lead-free directive”, it also the use of Lead, Mercury, Cadmium, Hexavalent chromium, Polybrominated biphenyls, Polybrominated diphenyl ether, Bis(2-ethylhexyl) phthalate, Butyl benzyl phthalate, Dibutyl phthalate, and Diisobutyl phthalate. If our group decides to be RoHS compliant, the most concerning substance to our design would be lead. Lead is considered to be the standard for solder, and is often found in solder and solder paste. RoHS defines a maximum allowed concentration of < 1000 parts per million by weight. Since we are not planning on shipping our product overseas, our group will likely not be striving towards being RoHS compliant. However, in the event that our product ever does became sold internationally, it is important to define the standard here for future reference.

**3.1.3 SCAA Water, Roasting, and Brewing Standards**

The Specialty Coffee Association of America (SCAA),a non-profit organization and world leader in developing coffee knowledge, has well defined standards which regulate the optimal ways to produce quality coffee. The Statistics & Standards Committee of the SCAA has defined the following water characteristics for optimal coffee flavor extraction and brewing. The water used in our coffee maker must meet the following specifications in order to ensure the best coffee possible. Refer to the table below for the complete list of water quality guidelines.

It is important that these guidelines be followed during the brewing process of our coffee to ensure customers who use our coffee machine get the best possible coffee quality that we can offer.

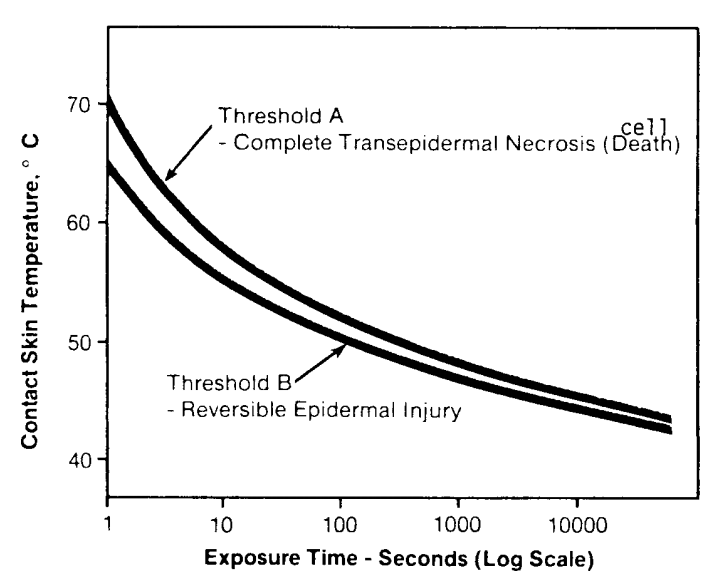
In addition to drinking water standards, the SCAA also has developed brewing standards which serve as guidelines on proper ratios and temperatures to make the ideal cup of coffee. Our project will follow the “Golden Cup Standard”, as defined by the SCAA. The “Golden Cup Standard” defines a coffee-to-water ratio of 55 grams per liter with a margin of error of 10%. The “Golden Cup Standard” also defines the optimal water temperature for the best flavor extraction from coffee grounds. Refer to Table 1 below for a summary of optimal water quality specifications.

*Table 1 -* Water Quality Specifications – Permission for reuse pending

It is recommended that the water temperature at the point of contact with the coffee to be 200°F ± 5°. The PID controller in our machine will be calibrated to maintain that temperature throughout the entire brewing process to guarantee users have the best possible coffee that we can offer. The final recommendation by the SCAA is that the container who receives the coffee must be able to maintain the coffee temperature between 176°F and 185°F, and at the no point must the temperature of the coffee increase due to contact with a heating element.

**3.1.4 ASTM Heated Surfaces Safety**

The American Society for Testing and Materials (ASTM) develops and defines standards for a wide range of materials and products. Our coffee machine must be compliant with ASTM code C1055-03, the standard guide for heated system surface conditions that produce contact burn injuries. According to ASTM temperatures in exceedance of 140°F have been shown to cause burn injuries, therefore our machine will take the necessary precautions to make sure it is safe and reliable for its users. Picture 14 below is illustrated in the pdf document by the ASTM titled “Standard Guide for Heated Surface Conditions that Produce Contact Burn Injuries”. In order to ensure the safety of the users using our coffee machine, our machine must be compliant with the ASTM standard.



*Figure 14* - Temperature-Time Relationships for Burns

**3.1.5 C Language standards**

The official ISO standard for the C programming language is ISO/IEC 9899:2011, or more commonly referred to as C11 [9].This document describes an overall conceptual view of how C programs should be prepared and interpreted by other developers, such as syntax and semantics rules. The standards for our design will be based off of ISO/IEC 9899:2011 as well as other conventional C program styling guidelines. The more notable standards to be considered are those that will directly addressed in this document. All of the system’s programs will have a consistent structure that is broken down into the following components.

* Brief - A small description of the program’s functionality and purpose of all the major components in the file.
* Header Files - This section will contain all imports, including custom made header files, with a comment for its purpose.
* Global - All global definitions are to be placed here.
* Functions - The code to be executed. Functions that are closely related should be placed in a proper order for which they would be executed.

Ample white space should be used in order to increase readability when reading through the code. At least three blank lines should be placed in between each function. And one blank line should between each function definition and the first line of code within a function. Braces will be required for all if, while, and do statements, and the ternary operator “?:” should be avoided if possible. This will keep all statements in a uniform manner to prevent the chance of confusion when scanning through the code.

Comments should be made often and as detailed as possible. Comments are a vital part to the programs in this system because they are the guideline for the code’s execution and help increase readability for a developer who did was not the source. A header and footer should be placed in every file and contain information about the files general purpose and any dependencies. Any blocks of code whose function is not immediately obvious to the reader should have a comment above it expressing the basic functionality. For every function a short brief must be commented out, above its definition, to describe its role. Any variable declarations should have what the variable is holding commented out next to it.

**3.1.6 JavaScript Language Standards**

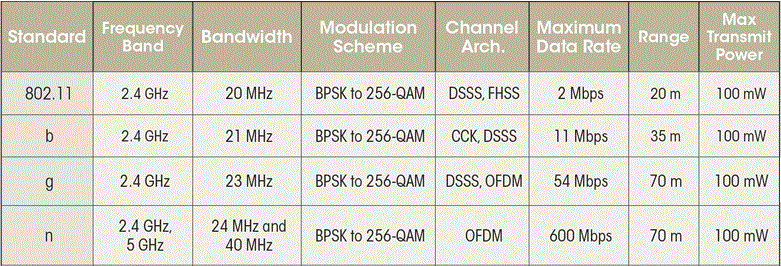
ECMAScript (or ES) is a trademarked scripting-language specification standardized by ECMA International in ECMA-262 and ISO/IEC 16262. It was created to standardize JavaScript, so as to promote multiple independent implementations. JavaScript has remained the best-known implementation of ECMAScript since the standard was first published, with other well-known implementations including JScript and ActionScript. ECMAScript is commonly used for client-side scripting on the World Wide Web, and it is increasingly being used for writing server applications and services using Node.js.

The 6th edition of ECMAScript, officially known as ECMAScript 2015 or ES6, was finalized in June 2015. This update adds significant new syntax for writing complex applications, including classes and modules, but defines them semantically in the same terms as ECMAScript 5 strict mode. Other new features include iterators and for/of loops, Python-style generators and generator expressions, arrow functions, binary data, typed arrays, collections (maps, sets and weak maps), promises, number and math enhancements, reflection, and proxies (metaprogramming for virtual objects and wrappers).

Since ES6 isn’t supported by most modern-day web browsers, in order to run the ES6 that will be used in the front-end of this application, the code must be transpiled to an older version of ECMAScript. This is the function of Babel and Webpack. Webpack is a module bundler. Its main purpose is to bundle JavaScript files for usage in a browser, yet it is also capable of transforming, bundling, or packaging just about any resource or asset.

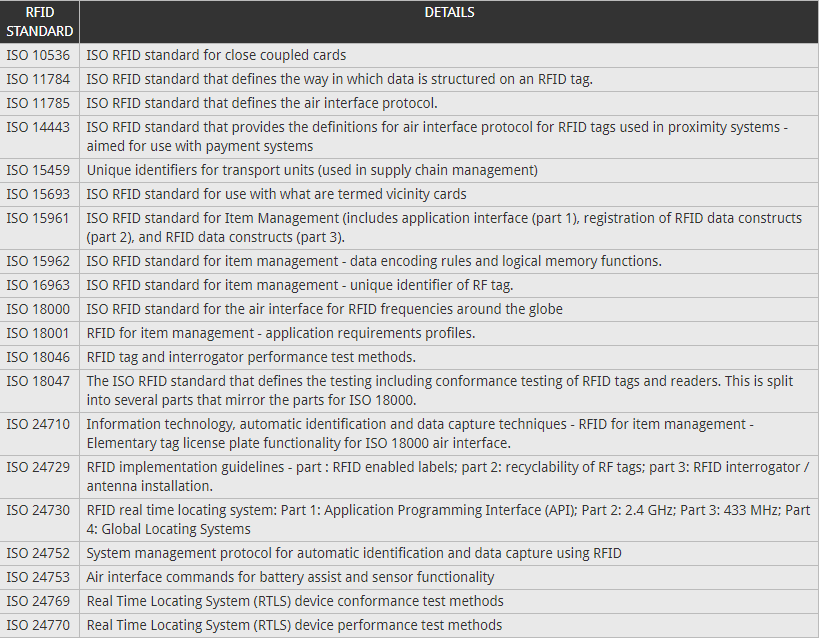
**3.1.7 Wi-Fi Standards**

Wi-Fi is the technology that allows personal computers, laptops, mobile phones, and other consumer electronics to connect to high speed internet without the need of a physical wired connection. The technology is based on the IEEE 802.11 standards. The standards define specifications for implementing (WLAN) communication in the 900MHz and 2.4, 3.6, 5, and 60 GHz frequencies. They are the most widely used computer networking standards, used in most consumer electronics which connect to the internet. This project will focus on IEEE 802.11b/g/n. The differences between the 3 revisions has been summarized in Table 2 below.

*Table 2 - Wi-Fi Standards - Permission Request Pending*

**3.1.8 RFID Constraints**

Radio-frequency identification (RFID) is a technology where digital data, encoded in RFID tags, are captured by a reader via radio waves. It is a similar technology to barcodes, in respect that data from a tag is captured by a scanner and is stored in a database. However RFID presents several advantages over barcoding. There are two main international RFID standardization bodies: the International Standards Organization (ISO), and the Electronics Product Code Global Incorporated (EPCglobal). ISO has defined different categories of RFID standards, these include: air interface and associated protocols, data content and formatting, conformance testing, applications, and several other smaller areas. Table 3 below summarizes the most important ISO RFID standards.

*Table 3 - RFID Standards – Permission Request Pending*

**3.1.9 Milk Storage Standards**

Moka will be incorporating a refrigeration sub-system that will be used to store and dispense milk for the user’s coffee. Without proper care milk can spoil very easily, so we will be referring to some guidelines set by the Western Dairy Association for proper care and storage of the milk in our system. According to the Western Dairy Association, milk must be stored 40 degrees Fahrenheit for proper safety and is best served between 35 to 40 degrees Fahrenheit. If the systems milk is exposed to prolonged temperatures above 40 degrees the milk in our machine will have to be exchanged to ensure quality and safety for our customers. At this temperature milk can remain safe for around seven days, so our system will need to have its store of milk replace before this point to ensure customer safety. The Western Dairy association also states that milk must be stored in a sanitary environment. Once the milk has left its container it must not be returned because it then risks the integrity of the whole container and could result in an early replacement of Moka’s milk storage. The system’s pump design will have to take this into consideration to keep the integrity of its milk storage for as long as possible.

**3.1.10 Git Standards**

Git will be our main tool in establishing a centralized source for the code of our system. To increase productivity and efficiency of the code’s development cycle, a centralized workflow must be established. In a centralized workflow, there is one central repository that everyone synchronizes there work into. Each developer will clone the repository and then check out a new branch for themselves to work on. Once they are done making changes they will then use git commands to commit there changes to the working branch and then push their changes to the GitHub server. After this a merge request will need to be created for the changes to be added to the master branch. A co-developer will need to go in and approve the changes before they are merged. This code review will prevent the amount of errors that are added to the master branch and increase the overall integrity of the code. This type of workflow can, especially with larger groups, can lead to a lot of discrepancies in each developer’s cloned repository as updates are merged into the master branch. To counteract this, out developers will be required to follow a set of steps before attempting to push any new changes to the server:

1. git checkout master
2. git pull
3. git checkout <working-branch>
4. git rebase master
5. git push origin <working-branch>

These steps make the developer update their cloned repository with any new changes that may have been added. Then the new branch needs to be rebased on top of master the branch. Rebasing is a little different than normal merges because instead of combining the two current branches with their most recent ancestor, it instead will replay all the changes of one branch onto the other. This will leave intact all commit messages for each step of the development, which results in a clearer commit history for code review. Once all of this is done, the developer is then free to push their changes to the server for review.

**3.2 Project Constraints**

The Accreditation Board for Engineering and Technology (ABET), is a non-governmental organization that defines standards and constraints to be followed by post-secondary education programs.Every engineering project must contain well defined limitations in its design in order to be successful. ABET defines constraints that could be economic, environmental, social, political, ethical, health and safety, manufacturability, and sustainability. The following section will go in detail into each of the ABET defined constraints, and how they may or may not be application to the design of our coffee maker.

**Economic** In order to make our product competitively priced, its total manufacturing price must not exceed $1,000. Similar products in the market range from $1,000 to $2,000. Early iterations of our machine included four different grinders who would grind different coffee beans on the spot based on input choices from the user. The cost of incorporating these separate coffee grinders into our system quickly became exorbitant and we were forced to scrap it from our design. The most optimal heating element for our project was an important design decision. The first important consideration we had to make was whether to use a heat exchange boiler as opposed to a thermoblock. Although the thermoblock is much cheaper, it takes longer to heat up water and is not able to handle the same quantity of water as a single boiler. After analyzing competing products in the market, we decided to opt for the more expensive option of a single boiler. Different kinds of boilers and their sizes were also factors we took into consideration when choosing our single boiler. There were also careful considerations in the choice of our microcontroller, and coffee ground dispenser, to find a balance between quality but economical design.

**Social** These constraints do not apply to our project.

**Political** These constraints do not apply to our project.

**Ethical** Concerns have been raised over the use of our integrated database to store our users’ drink preference and payment information. With recent worldwide scandals regarding private data leaks by giant technology corporations, such as Facebook and Cambridge Analytica, it is vital to have a system in place that will ensure our users’ privacy are uncompromised, and their data is securely stored. Safely storing payment information is of utmost importance in this project, therefore, to address these issues our coffee machine’s database will include SHA-256 encryption. SHA-256 is a cryptographic hash algorithm that generates a unique 32-byte signature for a text input. This will help ensure the payment information for the users of our coffee is securely encrypted and stored.

**Health and safety** Our machine should be safe to operate by any user, and serve a cup of coffee that will not cause any health problems to any user. As detailed in section 3.1.4 SCAA Water and Brewing Standards, there are limitations for what kind of water we must use in our machine in order to ensure it is safe for consumption and to be used to brew coffee.

**Sustainability** The goal of our product is to design a coffee machine that will be reliable long-term investment for its users. Aside from minor maintenance, such as refilling the water reservoir and cleaning, our machine is estimated to run without major maintenance for at least 5 years.

**Manufacturability** Our machine should be easy and convenient not only to install, but also to manufacture. The estimated dimension for our machine is 30 inches in height, 20 inches in width, and 25 inches in depth. This will give us plenty of space to house the internal components, while also remaining flexible and able to be accommodated in any regular coffee shop table.

**4 Research and Background Information**

The subsequent sections will contain all the research done to plan out all the different subsystems and components we will need to acquire to meet all of the system’s design requirements.

**4.1 Microcontroller Considerations**

The following sections include information on the top considerations for microcontrollers to be included in our system. Multiple microcontrollers were selected and researched in order to find the best fit for our system. The requirements considered in this search were the inclusion of either an integrated Wi-Fi module or a capability to communicate using an external Wi-Fi/Bluetooth module.

**4.1.1 Particle Photon (P0 Module)**

The Particle Photon is an inexpensive and small Wi-Fi Internet of Things(IoT) device that is powerful and easy to use. The Photon itself is a development board that would primarily be used during the testing phase and would not be used in the final rendition of our system. Although, the P0 Wi-Fi module on the board may be used in the final stages of the system. As seen from Table 4 below, the module is based off a STM32F205 microcontroller as its core with a power processing speed and plenty of flash and program memory. This module has an integrated single band 2.4GHz 802.11b/g/n Wi-Fi chip that can support up to 65 Mbit/s. This module supports all of the main Wi-Fi security modes so it should be able to run in most settings.

*Table 4 -* Particle P0 Module Basic Specifications

|  |  |
| --- | --- |
| **Specification** | **Value** |
| Processor | 120MHz - STM32F205 core |
| Memory | 128 KiB RAM  1MiB Flash Memory |
| Communication Methods | SPI  I2C  I2C  UART  802.11b/g/n |

**4.1.2 ESP32**

The ESP32 is a low cost IoT microchip that has built-in Wi-Fi and Bluetooth capabilities. The chips processor can either be a dual-core or single-core Tensilica Xtensa LX6 microprocessor and, as shown in table 5, can have a clock frequency of up to 240MHz. The chip’s Wi-Fi module contains an 802.11 chip @ 2.4GHz that can support data transfer at speeds up to 150 Mbit/s, which is almost three times what the particle photon can support. The chip’s Bluetooth module is compliant with Bluetooth v4.2 BR/EDR and BLE specifications and has UART HCI speeds of up to 4Mbps.

Like all the other chips that we have considered, the ESP32 is part of a series a low power chips. “It features all the state-of-the-art characteristics of low-power chips, including fine-grained clock gating, multiple power modes, and dynamic power scaling” (32), meaning it will only be useful for reducing our system’s overall power consumption. The different power modes would allow us to take advantage of a low power co-processor that is able to perform data computations and conversions even while the chip is in deep sleep. Refer to Table 5 below for the ESP32 basic specifications.

*Table 5 -* ESP32 Basic Specifications – [16]

|  |  |
| --- | --- |
| **Specification** | **Value** |
| Processor | 160-240MHz - 32-bit LX6 |
| Memory | ROM: 448 KiB  SRAM: 520 KiB  RTC fast SRAM: 8 KiB  RTC slow SRAM: 8 KiB  Embedded flash: Up to 4 MiB  External Flash: Up to 16 MiB |
| Communication Methods | SPI  I2C  I2S  UART  802.11b/g/n  Bluetooth |

The ESP32 has a numerous amount of advanced peripheral interfaces, including 34 programmable GPIO pins and can support CAN 2.0, that our system could take advantage of. While the ESP32 seems perfect for our system because it is cheap, low-power, and had integrated wireless communication, the main concern that led to this not being chosen as the main chip for our system is lack of documentation on some of its features. The chip is still currently targeted towards research developers and there are still some bugs within the chip that are being worked out and some of its peripheral features do not have extensive documentation.

**4.1.3 ESP8266/ATmega328P**

The ESP8266/ATmega328P combo was highly considered for use in the system. The ESP8266 it is a low-cost microchip with microcontroller capabilities and a full TCP/IP stack for Wi-Fi communication. From table 6 below the chip’s technical specifications are not as high as some of our other considerations. However, its main purpose in our system is to handle all of the networking applications for our system. The ESP8266 would have code flashed onto it that will deal with all of the incoming communications from the user’s app and use this information to then access our database to get the user’s information. This information will then be passed on to the so-called ‘brain’ of our machine ATmega328P.

*Table 6 -* ESP8266 Basic Specifications

|  |  |
| --- | --- |
| **Specification** | **Value** |
| Processor | 80MHz - L106 32-bit RISC |
| Memory | 32 KiB instruction RAM  32 KiB instruction Cache RAM  80 KiB user data RAM  16 KiB ETS system data RAM  1MB Flash Memory |
| Communication Methods | SPI  I2C  I2S  UART  802.11b/g/n |

The ATmega328P is a high-performance 8-bit AVR RISC-based microcontroller that can ideally achieve throughput of 1 MIPS per MHz. The ATmega328P will be responsible for controlling the rest of the system based on the network data transferred to it. From the data shown in table 7, the technical parametrics of the ATmega are low compared to every other microcontroller that has been considered for the system so far. Another downside of using this combination of these two chips over a more powerful integrated chip would be the modularity of the system’s functions. Their dependence on one another could increase the difficulty of debugging the overall system.

*Table 7 -* ATmega328P Basic Specifications – [4]

|  |  |
| --- | --- |
| **Specification** | **Value** |
| Processor | 20MHz - L106 32-bit RISC |
| Memory | 32 KiB Programmable Flash  1 KiB EEPROM  2 KiB SRAM |
| Communication Methods | SPI  I2C  UART |

**4.1.4** **Arduino MKR1000(SAMD21 Cortex-M0+)**

After all the considerations of the above microcontrollers, our main choice for the microcontroller is the Arduino MKR1000. This MKR1000 was designed to be a cost-effective solution for those seeking to create IoT projects with minimal previous networking experience. According to Arduino the chip is based on the Atmel ATSAMW25 which is composed of 3 main blocks:

* SAMD21 Cortex-M0+ 32bit low power ARM MCU
* WINC1500 low power 2.4GHz IEEE® 802.11 b/g/n Wi-Fi
* ECC508 Crypto Authentication

The SAMD21 is 32-bit low power MCU and as can be seen from Table 8, its processor may not be as powerful as some of the other Wi-Fi integrated MCUs we considered, but it should be more than enough to handle the functions of our and system. The rest of the technical specifications were about the same if not better, especially in terms of memory and available peripherals. The most important factor that made our decision was that this was a Wi-Fi integrated chip that would be the large amount of documentation and that it would be compatible with the Arduino libraries that will be used for some of Moka’s subsystems. Use of this MCU could benefit the development of the system in the prototyping phase of our design. The SAMD21 is the main microcontroller for the Arduino MKR1000 Wi-Fi dev board, so initial testing of the system can be done earlier using the Arduino before the microcontrollers are integrated into the PCB.

*Table 8 - SAMD21* – [5]

|  |  |
| --- | --- |
| **Specification** | **Value** |
| Processor | 48MHz – Cortex-M0+ 32bit ARM |
| Memory | 256 KiB Programmable Flash  32 KiB SRAM |
| Communication Methods | SPI  I2C  UART  802.11b/g/n |

**4.2 Serial Communication Technologies**

In this section, we describe the research done on different serial communication technologies. Before going into the details of why we chose to use the serial communication interface that we did we first have to talk about why it is that we need a serial communication interface in the first place and what a serial communication interface actually does.

A serial communication interface (SCI) is a protocol that enables the one bit at a time exchange of data between a microprocessor and peripherals such as printers, external drives, scanners, or mice. Serial communication interfaces also allow us to communicate with different sensors and motors.

A serial communication interface is ideal for our project as it allows us to communicate with the RFID module on our machine while keeping the amount of required GPIO pins on our microprocessor to a minimum.

**4.2.1 UART**

The serial communications interface that we will be utilizing with the RFID module is UART. takes bytes of data and transmits the individual bits in a sequential fashion. The UART takes bytes of data and transmits the individual bits in a sequential fashion. At the destination, a second UART re-assembles the bits into complete bytes. Each UART contains a shift register, which is the fundamental method of conversion between serial and parallel forms. Serial transmission of bits through a single wire or other medium is less costly than parallel transmission through multiple wires.

The data that we will be transmitting over the UART protocol is the unique ID of the scanned cup’s RFID tag. This ID will be used by the backend to determine which user is making the request and send back the appropriate data

**4.3 Wireless Communication Technologies**

Wireless communications discussed below are those that were considered for communication of an any external device to our system. Wireless communication for our system refers to the multiple ways for the user to interact with our system from an external source. Our system will also require wireless communication to contact our database servers to supply the user’s information.

**4.3.1 RFID**

Radio-frequency identification, or RFID for short, is the use of electromagnetic fields to track and identify objects that have a tag or label associated with the system. Some common uses of RFID systems today would be electronic toll collection on highways or tracking for warehouse storage. The two different components of the RFID system are the tags/labels and the reader itself. The tag receives a signal from the two-way RFID reader and sends back the information currently written into its memory. This is where the reader would send the data to a computer for the data to be used by the system that the reader is currently attached to.

There are currently three different types of tags: passive, active, and battery-assisted passive. A passive tag is the cheapest type of tag out of the three because they have no internal battery, so to transmit data they use the radio energy transmitted by the reader. The active tags have a built-in battery so it will periodically transmit its ID signal to RFID readers in the area. Battery assisted is a combination of both active and passive tags. The battery-assisted tags still power themselves through an internal battery. However, they wait until they are within range of a reader.

There are two separate types of readers: active and passive. Active readers periodically send out interrogator signals waiting for a response from a tag. Unlike active readers, passive readers await the signal of an active tag instead of sending out a signal of its own to search for tags. The combination of these different active and passive components can be combined into different RFID systems such as the Active Reader Passive Tag system that simply uses an active reader to search out and get information from passive tags [28].

There are six main frequency ranges that the RFID system transmits data over. These frequencies and their specifications can be seen in Table 9 below. Our system will more than likely implement an RFID system on the 120-150 kHz frequency ranges because this is a short-range frequency band that is unregulated and usually used in smaller applications such as animal tracking. This is unlike the 13.56 MHz band which falls under ISM band regulations. ISM bands are usually reserved for industrial, scientific and medical purposes.

*Table 9 -* Basic RFID Frequency Info – [28]

|  |  |  |  |
| --- | --- | --- | --- |
| **Band** | **Regulation** | **Range** | **Data Speed** |
| 120–150 kHz (LF) | Unregulated | 10 cm | Low |
| 13.56 MHz (HF) | ISM band  worldwide | 10 cm–1 m | Low to moderate |
| 433 MHz (UHF) | Short range devices | 1–100 m | Moderate |
| 865-868 MHz (Europe)  902-928 MHz (North America) UHF | ISM band | 1–12 m | Moderate to high |
| 2450-5800 MHz ([microwave](https://en.wikipedia.org/wiki/Microwave)) | ISM band | 1–2 m | High |
| 3.1–10 GHz (microwave) | Ultra wide band | Up to 200 m | High |

**4.3.2 Bluetooth**

Bluetooth is a form of wireless communication that allows paired devices to communicate in an ad hoc manner by building personal access networks. Ad hoc wireless communication involves direct communication between the devices in question and avoids the need for an access point to connect multiple Bluetooth enabled devices. Bluetooth devices exchange data over short distances from a fixed device and a personal area network that operates on 79 separate channels on a UHF radio band frequency of 2.4 - 2.485 GHz (CITE). When data is being transmitted it is done through the method of Frequency-hopping spread spectrum (FHSS). FHSS is a method of transmitting radio signals by rapidly switching between channels that are not occupied. This method is advantageous because FHSS signals are harder to intercept and allows the devices to share a frequency band with many types of other transmissions with minimal interference.

One undesirable trait of Bluetooth that could be a problem for our system would be the need for the communicating devices to be paired with one another. There are two methods of pairing devices in order for them to communicate. The old method called Legacy pairing where each device must enter a PIN code and they will only be paired if the PIN codes match. Modern forms of Bluetooth now use Secure Simple Pairing (SSP), that involves a public key cryptography to pair devices(wiki). This improvement helps make communication more secure and enables more types of devices to be able to be paired, but for our design, this could still cause problems since Moka will be placed in a commercial setting.

**4.3.3 Wi-Fi**

Wi-Fi is another form of wireless communication that allows devices to connect to each other through two-way radio communication. The Wi-Fi branch was developed by an organization called Wi-Fi Alliance whose technology allows devices to connect to each other through Wireless Local Area Network(WLAN) to communicate with each other under the IEEE 802.11 standards. These 802.11 standards are a set of media access control and physical layer specifications for implementing wireless network communication. The current standards allow for devices to transmit over the band frequencies of 2.4 GHz and 5GHz. These frequencies than normal radio communication to accommodate for larger data transfers.

In a normal infrastructure setup, devices that have Wi-Fi communication capabilities connect to a WLAN through some access point, most commonly a router, with the same SSID. These devices transmit its data in the form of a radio signal to a wireless router. This router then decodes this information and sends it to the internet through a wired connection that is tied to the router. When a device is trying to receive data, the process described above is done in reverse.

With all considered, Wi-Fi will be the form of wireless communication that will be best suited for our Moka implementation. While Bluetooth may have lower power consumption and cost, it has a much shorter range and data transfer rate. Cost and power consumption are both essential requirements for our system; the tradeoff of a greater network speed and range at which the customers can order their drinks is more desirable for our Moka implementation. With the integration of our mobile app, our system would need to be able to transmit data over larger ranges than what a Bluetooth PAN can provide.

**4.4 App Development**

In the following sections we will be exploring the different types of research we did for the frontend and backend development of the user application.

**4.4.1 Frontend**

We had many different routes we could have taken for the frontend of the mobile application. Before deciding on which frontend framework or platform we were going to develop for, we needed to answer a few questions. Which devices do we want the application available? Do we want this to be a web application or a native device application? Which development environments do we need support for? In this section we will be exploring the different options that we were presented with when thinking of how to develop the user application.

For the user application to be effective in facilitating the ordering process for the coffee, it would have to be available for all the major mobile operating systems. Since the majority of the mobile operating system today is dominated by iOS and Android we needed to ensure compatibility for these two operating systems.

The first option available for creating an application that would be compatible with all operating systems was to create a web application. A web application usually consists of a client and a server where the client displays the information given to it by the server. The client in a web app is the user’s web browser. Since all modern smartphones have a web browser, a web application is compatible with all smartphones.

There are many different frontend web application frameworks to choose from but they are basically divided into two sections, PHP frontend frameworks and JavaScript frontend frameworks. When it comes to PHP frontend frameworks one can either go with a full-fledged web application framework like Laravel or utilize a content management system such as WordPress to achieve similar results. Since we had experience utilizing both frameworks, we restricted our research to only these two PHP options.

First, we found we determined what exactly Laravel is. Laravel is an open-source PHP web framework intended for the development of web applications following the model view controller architectural pattern and based on Symfony.

Some of the features of Laravel are:

* modular packaging system with a dedicated dependency manager
* different ways for accessing relational databases through Routing
* utilities that aid in application deployment and maintenance
* orientation toward syntactic sugar
* easy authentication by providing a simple & easy to use interface

Next, we did some research on WordPress. WordPress is a free content management system based on PHP and MySQL. To function, WordPress has to be installed on a web server, which would either be part of an Internet hosting service or a network host in its own right. WordPress’s primary function is to create blogs, however, through the use of plugins and themes, WordPress could be configured to act as a sort of web application.

We concluded that if we decided to go with a PHP frontend framework, we would choose Laravel as it is lightweight and gives us the routing and request options that match with our backend.

The next area of research was JavaScript frontend frameworks. There are many free open source JavaScript frameworks that would be able to create the web application we need. The main JavaScript frontend frameworks that we looked into were, AngularJS and React. In the coming sections, we explore these three JavaScript frontend frameworks to determine which one would be the best fit for or our web application.

AngularJS lets you write web applications by expanding HTML. It lets you use HTML as your template language and lets you extend HTML’s syntax to express your application’s components. It automatically synchronizes data from your UI with your JavaScript objects through 2-way data binding.

React makes it painless to create interactive UIs. With React you design components for each module in your application. When the data for the application changes, React will only re-render the components that are affected by those changes. This makes React very efficient when creating a web application with multiple views.

Although both of these JavaScript frontend frameworks would work, If we were to create a web app and use a JavaScript frontend framework, we decided we would go with React. React allows us to utilize the asynchronous AJAX requests that we need to make to retrieve information from the backend and then seamlessly render it to the browser without the need for any browser refreshes thus resulting a great user experience.

Although a web app would work well for our user application, we could also use a native development platform such as Swift for iOS and Java for Android. The problem with choosing one of these native development platforms is that we would have to essentially develop the application twice, once for android and one for iOS. Alternatively, we could use something like React Native which allows us to develop the application in JavaScript and then compile it to each individual mobile platform.

React Native lets you build mobile apps using only JavaScript. It uses the same design as React for the web browser, letting you compose a rich mobile native application. Unlike other platforms such as Phonegap, React Native doesn’t create a mobile web application, it creates a real native application that is indistinguishable from an application built in Swift or Java.

Since we don’t have the need for a desktop available component to our user application, we decided that we will be utilizing React Native as the platform on which to build it. Using React Native will allow the user to always have their coffee preferences just a few taps away right on their smartphone home screen.

**4.4.2 Backend/Database**

Similarly, to the frontend, there were many different options and paths presented to us when developing the backend. The only thing that we knew for sure was that we wanted to build a modular backend that could be accessed by other developers or applications, that way we could build the frontend independently from the backend. The most efficient way to create such an interface would be to build an RESTful API (Application Processing Interface).

A RESTful API is a program that is executed on a server and listens to requests made to it and responds with formatted data from the system. These requests are made to the server via HTTP protocol to the server’s IP address. The data the server responds with could be anything but usually, the server responds with data from a local database. This could be user information, a list of phone numbers, or map information. Basically, the job of the backend server is to retrieve the data from a database and format the data so that it can be parsed easily by whoever is requesting it.

Now that we have clearly defined what kind of backend we want to develop, we will be exploring the different languages, libraries, and databases we could potentially use and also explore which use case each of those options would best serve.

Since the database operates independently from the backend server that we are developing, we will be discussing the different types of databases, which one we chose, and why we chose it.

There are two main types of databases, SQL databases and NoSQL databases. As the name implies, SQL databases are databases that utilize the SQL querying language to retrieve data. These databases are usually structured with a database having tables and those tables having rows of data where each row has many columns. The columns can be seen as key value pairs, the key being the column and the value being what is stored for that given entry in its row. These databases are usually also referred to as relational databases.

NoSQL databases on the other hand, provide a mechanism for storage and retrieval of data that is modeled differently than the tabular relations used in relational databases. The idea of NoSQL databases was motivated by the idea that relational databases were not designed to cope with the scale and agility challenges that face modern applications. They were also not built to take advantage of the commodity storage and processing power available today. For this reason, in most cases a NoSQL database could outperform a relational database.

Based on the research we conducted, we chose to use a NoSQL database, MongoDB, to be the database for our project. MongoDB stores all of its data in JSON format which is ideal for our project because it is what we plan to format the response data from our backend server. MongoDB is also quick and flexible allowing us to define dynamic schemas for each of our models without the need of creating many different tables.

Next we will be discussing the different types of API development platforms and the different frameworks available on each. Similarly to a frontend web application, the backend servers are mainly developed utilizing two different languages, PHP and Javascript. PHP based backends are executed by the PHP engine which is installed on the server. Javascript based backends run on an engine called Nodejs. Nodejs is essentially the Javascript V8 engine ripped out of the browser.

First, we will be exploring the different options for developing a RESTful API in PHP. The PHP engine is installed on the host server using its respective package manager. The next thing to decide after the engine is installed is which framework to utilize for building the API. The two frameworks that we will be researching are Laravel and Slim.

Laravel is extremely configurable and extendable. Due to the fact that it’s open source, free, and very popular, Laravel has a variety of derivatives, forks, manipulations, and extensions that allow the developer to make it do whatever they want it to do. One of the best features of Laravel is that it supports PHP 7 which is very good because PHP 7 has many enhancements that increase general performance across the board.

Slim's documentation is well organized and detailed, every concept is thoroughly explained and it is very helpful for both advanced users and beginners. Slim also supports code hooks for executing functions at different points in time during the application's lifecycle.

Next, we will be exploring the different options for developing a RESTful API using Nodejs. Similarly to PHP, the first thing the developer must do to utilize Nodejs is install it on the host machine. Once installed we have do decide which framework use. The two frameworks that we will be comparing are Express and Meteor.

Express is a minimal and flexible Node.js web application framework that provides a lot of features for web and mobile applications. It allows you to easily define routes that can deliver data in many different formats. Express also allows you to create middleware that runs in between every route allowing for trivial authentication implementation.

Meteor uses something called “data on the wire” which means that Meteor doesn’t actually render data itself, instead, it chooses to transmit data for conversion and rendering on the client making it extremely extensible and flexible. One big downside to meteor is that it doesn’t allow for server side view rendering. This means that it is only used to transmit data and not for transmitting fully rendered views.

After conducting all this research on which backend system to choose, we decided that we were going to use Nodejs and Express to develop our backend.

Nodejs and Express allow us to use the same language for both the backend and frontend. Express also one of the most supported Node.js frameworks. It has an open-source community, so the code is always reviewed and improved.

**4.5 Power Technologies**

Powering the coffee machine was arguably the most vital aspect of the entire project. Without an adequate power supply supplying the correct voltages and currents to the electrical components will not only cause them to not perform properly, but it could also prove to pose even a hazardous working condition for the group members. The main purpose of a power supply is to provide the correct current, voltage and frequency to its load. The clear majority of consumer electronics require a DC input voltage to operate. Regulated power supplies are regular power supplies with a voltage regulating component. The AC mains input is fed to the power supply, and this is output is in turn fed to the voltage regulator to provide the final output. The final output voltage will then remain stable for use in consumer electronics and components, regardless of variations in the AC input voltage or load requirement. The following section will detail different kinds of power supplies, how it relates to our application, and considerations on on the most optimal choices for our project.

**4.5.1 Power Supplies**

Switched-mode power supplies (SMPSU), employ switching regulators, such as MOSFET, which continuously turn on and off at high frequencies in order to efficiently convert electrical power. The high frequencies allow for the use of small transformers and filters, therefore making it more practical. SMPSU’s also employ feedback control loops to regulate and stabilize output voltage. In this type of power supply, the AC input from the mains is immediately rectified and then and filtered to obtain a DC voltage. The resulting DC voltage is switched on and off using MOSFET at frequencies ranging from 10 kHz to 1 MHz. This high frequency AC output then passes through a transformer where it is lowered to the specified voltage of the power supply. However, this resulting output is still AC and therefore needs to be filtered and converted between it can be used in regular consumer electronics and components. The diagram below illustrates the process by which SMPSU’s convert AC input mains to usable DC output, as well as the feedback control loop used to stabilize the DC output.

Advantages of switched-mode power supplies include:

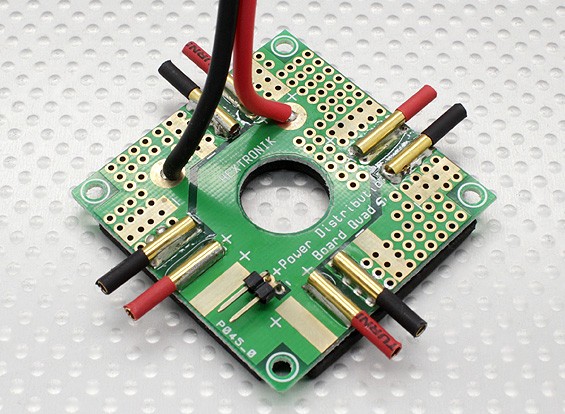
* More compact due to use of smaller transformers, therefore they are able to be used in more consumer applications.
* Efficiency around 68% to 90%.
* Flexible technology.
* The transformer-isolated supplies have stable outputs independent of the input supply voltage.
* High power density.

Disadvantages of switched-mode power supplies include:

* Extra external components require them to use more space.
* Higher susceptibility to electrical noise.
* Complex design.
* Higher cost due to extra components.

**4.5.2 Power Distribution Board**

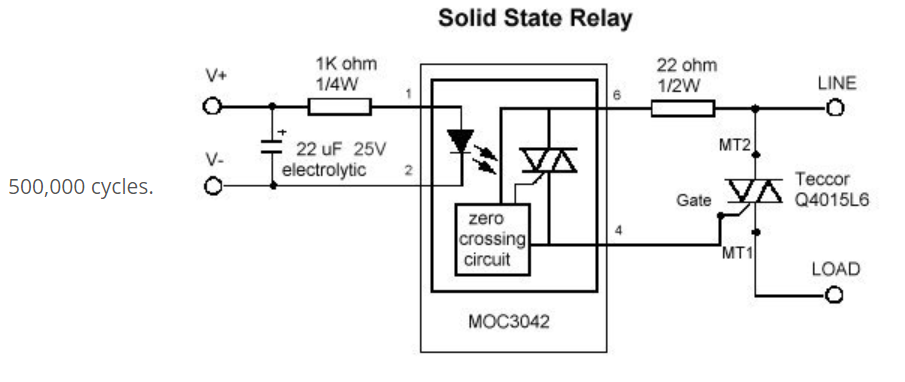
Our group had considered a power distribution board (PDB) in our project. PDB are fairly simple circuits, however, they provide a nice and effective way to connect all the grounds to each other and all the positive connectors together, eliminating the need of having messy power cables between all our internal components. PDB may also include voltage regulators within them. This is very useful for converting the voltage from our power supply to the correct voltages to power our PCB. Figure 15 below shows a sample PDB which our group has considered using in our project.



*Figure 15 -* Example of a Power Distribution Board

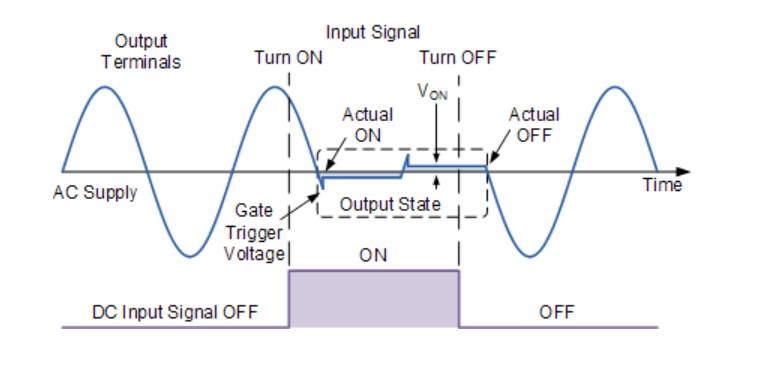
**4.5.3 Solid State Relays**

Solid-state relays (SSR), are electronic switching devices who turn on or off based on a small input voltage applied across its input terminals. They perform the same tasks as electromechanical relays (EMR), however, they do not contain any moving parts within it. This allows SSR’s to achieve faster switching speeds, as well as having no physical parts to degrade over time. A typical SSR contains optical semiconductors, such as photocouplers, which isolate the external input voltage signal by converting it into an optical signal. When an input voltage higher than the specified pick-up voltage is applied the circuit is turned and, and will be deactivated when the input is less than the required minimum voltage. Refer to figure 16 below for a circuit schematic of the MOC3042 SSR.



*Figure 16 -* Solid State Relay Circuit Schematic

The most common SSR’s are zero switching relays. These SSR’s will turn on the load when the external input voltage is applied and the load voltage crosses the zero point of the AC sine wave. The SSR turns off the load when the external input voltage is removed. As illustrated in figure 17 below, no matter where in the sinusoidal waveform, either in a positive or negative cycle, upon applying an external DC input voltage signal, the output will only turn on when the waveform crosses over the zero point.

*Figure 17 - Illustration of the zero point crossover of a SSR*

Advantages of including a SSR in our design over EMR include:

* Smaller and slimmer.
* Completely silent.
* Switching time is dependent on the time required to power the LED on and off, usually milliseconds.
* Increased lifetime due to the absence of not having moving parts, and no contacts to build up carbon.
* Constant output resistance.
* Less sensitive to storage and operating environment factors.

Resistive loads are the simplest application of a SSR. Keeping thermal considerations in mind, using an SSR will be ideal to switching the boiler on and off in our coffee maker.

**4.6 Software Tools**

Clear and efficient team collaboration was one of the key elements to expediting the process of creating a successful design. The following tools were chosen with this purpose in mind. The ability for reliable communication, file sharing, cloud storage, and the team’s familiarity were the main factors that were evaluated when looking into the tools our team would use for collaboration.

**4.6.1 Communication**

Communication is a key component of any design process that involves a team of designers. Reliable communication between all the designers to designate tasks and share information help establish and complete the goals of each stage of the system’s development.

Discord

The tool that was chosen for general communication between the team is Discord. The tool is an open-source software created by the company Discord. Discord offers secure server-based text and voice chat that can also be used to share small files between the team. Once a server has been created through the desktop or mobile app, the owner can then send out invite links to all necessary personnel. Within each server multiple channels can be created in order to organize the information that is passed between each team. Each subsystem of the project can be designated its own channel to decrease any confusion that could occur with having everything in one general chat. Within these channels permissions can be set that would allow the project owner to limit the use of channels to only those who require them if needed. Discord also features a full mobile application that works on both iOS and Android and make it easier for our team to remain up to date on the current development operations. These features combined with the team’s familiarity with the tool supported the decision Discord to be chosen as the main medium of communication for our team.

Google Drive

Google Drive will be utilized for more general record keeping and for file transfers of sizes that discord cannot handle. Google Drive is cloud storage system that offers up to 15GB of free storage space to those with a google account. This provides our team with plenty of room to keep all general documents in a place that is easily accessible by all designers. The majority of documents that will be stored here will be spreadsheets containing budget information and division of labor. The main benefit to utilizing Google Drive for these documents is because of the synchronization of work that allows the whole team to be able to access and edit the documents at the same time.

GitHub

GitHub is a tool that provided us with an efficient method for collaboration and peer review of all code written for the system. GitHub will allow our team to have a centralized source for our team to commit their git branches onto a master branch. GitHub Education provides a student developer pack that will give us access to many different tools to utilize. One of these benefits we will we using is the access to a GitHub Developer account which allows us to create private repositories to securely store all code and only allow those with a need to access and view our repository. The student developer pack also provides us with Git Kraken Pro which is another tool to track all branches and commits to our repository. Git Kraken will provide the design team with a clear visualization of all branches and commits that are created by anyone with access to the repository. This will help with the debugging process in the case that a bad commit has been made, or for any other complications that could occur in the code’s development cycle.

Trello

Assigning tasks and staying on a development schedule are going to be an important factor for our team to complete Moka. The tool Trello provided by Trello, Inc will provide our team with a virtual task board to work off. The project manager can create a board and then invite the rest of the team to gain access the board. With this tool all users will be able to create cards for any task that needs to be completed. These cards can be classified and store of in a collection of lists that can keep track of what stage of development that task is in; such as In-Progress or Completed. Each of these cards will contain detailed information on each task, such as deadlines and designers responsible for its completion. Trello will give the whole team a good perspective on the system’s overall development, which is why it was chosen to be the main tool for task management.

**4.6.2 Development**

During development, we utilized many different tools to aid with writing code, version control, database management, etc. Table 10 details how each software tool was utilized. This tools were integral to the overall development so that all of our separate developers could collaborate over a common workflow to get the design completed as quickly as possible.

*Table 10 - Development Tools*

|  |  |
| --- | --- |
| Tool | Use |
| Sublime Text | Programming user application |
| Atom | Programming user application |
| Arduino IDE | Programming microcontroller |
| Git | Version control software |
| GitHub | Uploading and keeping up to date version control history between all developers |
| Postman | Develop and test RESTful API routes to ensure correct data is being sent back |
| RoboMongo 3T | Manipulating, and testing MongoDB database to ensure correct data is being stored |
| Sketch | Developing User Interface and User Experience designs |
| Docker | Creating a consistent development environment between all developers |
| GitKraken | Ensure git is being utilized correctly |
| Google Chrome | View backend data |
| Node Package Manager | Handle dependencies for both the frontend and backend of the user application |

**4.6.3 Documentation**

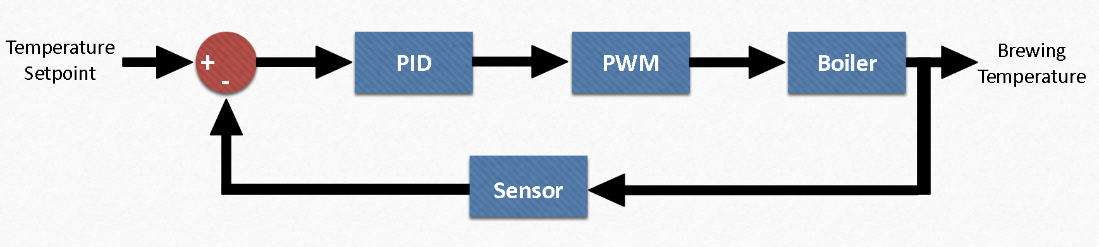
OneDrive – Microsoft Word

Creating a single unified document with a team of developers can be a difficult task with a lot of manual integration of work. To keep the manual integration and formatting fixes to a minimum the team decided to use a cloud file sharing service to store the master copy of all essential documentation. With Microsoft’s OneDrive our team will be able to synchronously update any documentation in real-time. This allows all designers to have clear perspective on the progress of the completion of the system’s documentation. The main reason OneDrive was chosen over many other cloud based storage systems is because it integrates the use of Microsoft’s sophisticated tools like Word and PowerPoint. OneDrive allows the use of online versions of Word and PowerPoint. This gives us the sophisticated formatting and tools of Microsoft Office while giving all team members the access to update any documentation in real-time.

**4.7 PID Controller**

PID control (Proportional-Integral-Derivative) is an algorithm used extensively in industrial control applications. In 1911, Elmer Sperry developed the first ever PID controller to automate a ship’s steering mechanism [24]. Today, this simple PID algorithm is used for automating processes ranging from furnace temperature control to neutralizing pH levels.

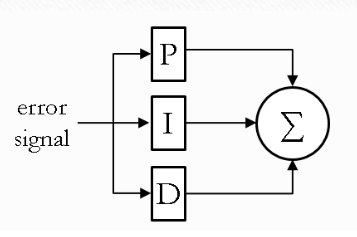
A PID controller is an implementation of this algorithm that adjusts for load changes by comparing the latter against a reference point. A process variable can be any measurable variable that is controlled such as flow, temperature, pressure, level, etc. The control loop feedback mechanism is displayed in Figure 18. The PID controller has two inputs, a set point and the sensor’s output, also known as the measured process variable. Essentially, the controller finds the difference between these two inputs and adjusts accordingly. This output informs the system how much adjustment is required to eliminate this difference through a PID algorithm. This output is converted into a pulse width modulated wave that can be used in conjunction with a solid state relay in order to power a large load. This process continues repeating until the system is turned off.



*Figure 18 -* Closed Loop Diagram of PID

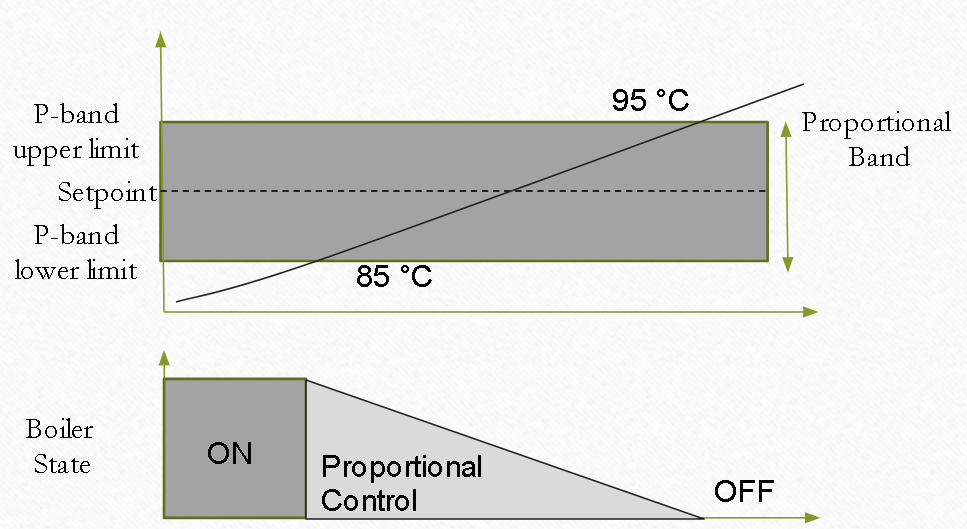
The popularity of the PID controller can be attributed to its high degree of performance and simplicity of operation. It has the ability to adjust the output’s state to anywhere from 0 to 100% of available power. The alternative to a PID is a controller that is either on or off, which causes an undesired ripple at the output. This is adequate for applications where stability is not a concern or in cases where the response delay and maximum change of rise is small; otherwise, a PID controller should be used to achieve completely stable control.

The PID controller block consists of three parameters that can be adjusted to achieve a desired output. These parameters are the proportional, integral, and derivative. Figure 19 illustrates how a PID controller adds its terms together.



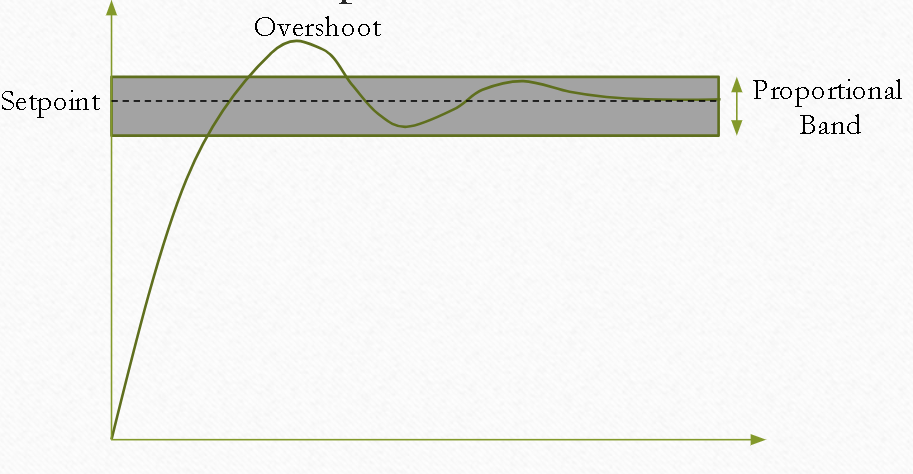
*Figure 19 -* PID diagram with gain terms

The proportional element of the controller simply changes the output in proportion to the error. A throttling range or proportional band is the amount of change in the controlled variable that drives the output from 0% to 100%. The controller behaves as an on-off controller outside this range. Typically, the set point is when the output is at 50%. Refer to figure 20 for an explanation of the proportional band.



*Figure 20 -* Proportional Band Explanation

If configured correctly, the proportional part of the controller will result in a stable output with a small offset compared to the set value. If the proportional band is too small, the controller will behave as an on-off controller and the output will continue to oscillate continuously. If the proportional band is too narrow, the set point temperature is never reached. Figure 21 below illustrates a correctly configured proportional-band.



*Figure 21 -* Correctly Configured Proportional-Band

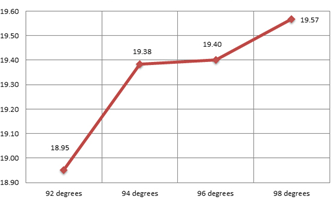
The integrator integrates voltage as a function of time. This allows the controller to reduce the offset from the proportional portion of the controller in integral time until the output matches the set point. A shorter integral time will eliminate the offset quickly. An integral time that is too short will result in overshoot that takes time to correct. An integral time that is too large is avoided because offset should be eliminated quickly.

The derivative element of the controller senses the change in error over a short time period and adds a portion of the error to the output. This is also called the disturbance action because it is the part of the controller that moves the output back to the set point in the case of an unexpected disturbance. In other words, it keeps the output within the proportional band if there is a disturbance. Similarly to the integral time, the derivative time is the time it takes to reduce the output capacity to its normal value. This leads to excessive response that takes longer to become constant. If it is too short, the controller will behave as a PI controller that does not try to correct disturbances

**4.7.1 Why PID**

The design will implement a PID controller to regulate the temperature of the boiler. Depending on the type of coffee desired, regulating temperature at a particular level yields an optimal cup of coffee. As mentioned previously, a PID is able to do this by calculating the error between the reference point and a measured variable. In the case of the boiler, the set point would be the desired boiler temperature and the measured process variable would be the temperature being measured by a thermocouple. Household coffee makers and some commercial ones do not include this technology. Instead, these units turn on the heating element when the water is below the desired temperature and turn the heating element off if the water is above it. This is not the most efficient way to heat water nor does it result in a quality cup of coffee.

The quality of a cup of coffee depends on factors such as the coffee-to-water ratio, quality of the water, and ground quality. One factor that is often overlooked by consumers is temperature. In fact, many coffee machines use a PID controller to brew coffee at a temperature ranging from 195 F (91 C) to 205 F (96 C). A higher temperature will cause over-extraction, resulting in a bitter and burnt-tasting coffee [6]. A lower one will cause under-extraction, which results in a sour, diluted, and weak-tasting coffee. For darker roast, it is recommended to brew on the higher end of this range while a light roast requires hotter brewing. Figure 4.9-5 shows the relationship between brew temperature and extraction yield percentage from a sample size of 6 cups. A difference of around 1% greatly alters the taste of coffee. Unfortunately, the vast majority of drip coffee machines (except ones certified by the SCAA) never reach adequate brewing temperature [14]. Figure 22 below illustrates extraction yield relative to brewing temperature in espresso.



*Figure 22 -* Average extraction yield relative to brewing temperature in Espresso

One of the principal ways water temperature can change when making coffee is through external disturbances. For example, opening of a door during winter could decrease the temperature of water while a sudden power supply surge could increase it. It is also important to note that different materials inside the coffee machine will have different thermal properties that affect temperature. These properties do not need to be accounted for since they will not change.

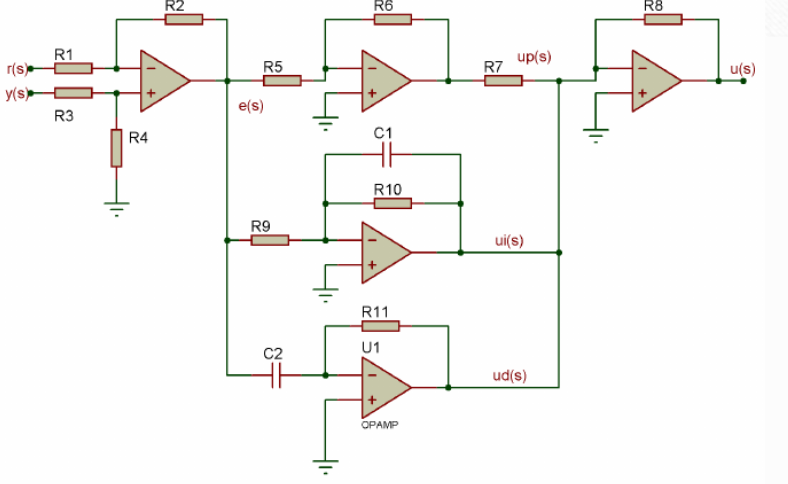
All of the previously mentioned factors prove that it is vital to have precise control over water temperature when making coffee. A PID controller has been shown to be the best way to achieve this for our application due to its precise output response. Many different implementations of the PID algorithm exist. Choosing the right one and implementing it appropriately will play a big role in the quality of the coffee.

**4.7.2 Parallel Op Amp PID Controller**

The three elements in a PID controller can be modeled by the mathematical functions of multiplication, integration, and differentiation.



Similarly, these same functions can be modeled in analog electronics by using operational amplifiers. An analog implementation of a PID controller is shown in the figure below. The proportional term can be modeled by an inverting amplifier since its transfer function is simply multiplication by the ratio of the two resistors. The integral and differential elements in the PID controller can be modeled by the integrator and differentiator operational amplifier configurations. Each of these operational amplifiers is added by the summing amplifier configuration. Although the first three transfer functions are negative, the fact that the summing amplifier transfer function is also negative reverses the output’s sign. Each of these parameters can be tuned appropriately depending on the values of the components. In the ideal case, the PID controller can be configured with the kp gain alone. Since it is more practical to tune each gain individually, the second form of the equation is the one that will be implemented [22]. Refer to figure 23 for analog implementation of a PID controller using op-amps.



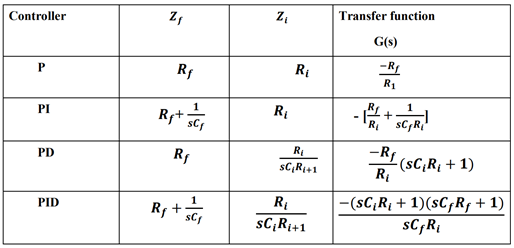
*Figure 23 -* Analog Implementation of a PID controller using operational amplifiers

The transfer functions for the controller are given below. As shown in table 11 below, this implementation allows for simple manipulation of the controller. For example, the derivative part can be removed to make it a PI controller instead.

By manipulating the PID transfer function, it is possible to obtain the following equation:



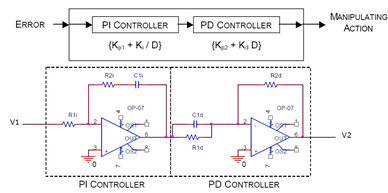
*Table 11 -* Controller Transfer Functions



**4.7.3 Series Op Amp PID Controller**

An alternative to the parallel PID controller is the interacting or series Op-amp based PID controller. The design of individual P, I, and D term do not change. As in the case of the ideal PID controller, the proportional action affects all three actions. Also, the Kp action is dependent on the derivative and integral tuning parameters. The accuracy of a practical PID controller would suffer using this design because this model assumes the parameters change linearly in real life.

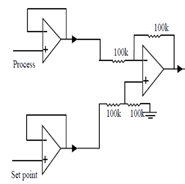




*Figure 24* - Series PID op-amp implementation

**4.7.4 Difference Amplifier**

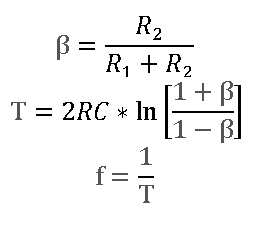
As shown in Figure 25, both PID implementations still need a way to find the error between the measured process variable and the set point. This can be done with the difference amplifier pictured below. This circuit finds the difference between the setpoint temperature and the actual temperature of the load device and amplifies this difference while suppressing the voltage that is common at both inputs. Hence, the output voltage equation is simply the gain of the amplifier multiplied by the difference between the non-inverting terminal and the inverting terminal. Its gain is given by dividing the feedback resistor by the resistor at the inverting terminal. An alternative to this circuit design for applications where amplifying a very small differential signal is needed, an instrumentation amplifier could be used instead.



*Figure 25 -* Difference Amplifier for op-amp PID

**4.7.5 Op-amp Astable Multivibrator**

Astable multivibrators switch continuously between two states without any external control. It can be designed through several methods such as by using an NE555 timer or by using logic gates. An op-amp astable multivibrator is a circuit that uses RC feedback to produce its own input. Comparing it to the alternatives, it gives a symmetrical output and works in the analog domain. The amount of time spent in between states is determined by the amount of time the capacitor is charging and discharging. The operational amplifier works as a comparator, which produces a rectangular waveform. The output is low if the input voltage is less than the voltage at the reference point and it outputs high if the input voltage is more than the voltage at the reference point. At first, the capacitor is fully discharged. Then it starts charging depending on the circuit’s RC time constant. Once the voltage at the inverting terminal is equal to the voltage a the non-inverting terminal, the capacitor begins to discharge at the same rate because the voltage across the capacitor is negative. The equations to configure the circuit are provided below. The duty cycle is controlled by changing the ratio of the resistors. Figure 26 below illustrates an astable multivibrator using op-amp.



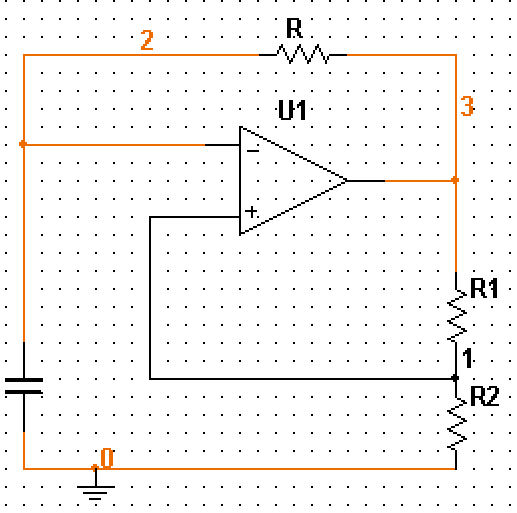


Figure 26 - Astable Multivibrator using Op-amp

**4.7.6 Power Amplifier**

Power amplifiers are needed to drive large output loads. They usually require the use of heat sinks due to the large amount of heat they dissipate.

A-Type Power Amplifier: The entire input is used in the amplification process. These is the simplest type of power amplifier because it only uses a single transistor. A disadvantage is that the transistor dissipates a lot of heat because it is always on. They have low distortion levels so they can be used at higher frequencies.

B-Type Power Amplifier: Uses two transistors to fix efficiency and heating issues associated with the A-type power amplifier. One transistor amplifies the positive half of the input and the second the negative half. Superposition causes a small distortion at the crossover region. To fix this issue, AB-type power amplifiers combine A and B amplifiers. This configuration requires the use of diodes and resistors to reduce bias voltage near the crossover region. This reduces the amplifier’s efficiency.

C-Type Power Amplifier: Allows for greater efficiency but has greater distortion due to a smaller conduction angle. Used mainly in high frequency oscillators and RF signals

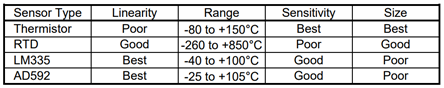
Power amplifier classes D, E, F, G etc. are called switching power amplifiers because they are used to amplify PWM digital signals. They can only turn the output to constantly ON or constantly OFF without any levels in between. Their advantage over the A, B, C power amplifiers is that they can achieve much higher power efficiencies.

An alternative to using power amplifiers is to use Darlington transistors. They come in a single package and behave as a normal transistor, but they have a very high gain. This means that they can increase current to drive a large load such as a motor. Power dissipation can be between 200 mW and 250 W. The type of transistors used is also an important factor affecting the power dissipated.

**4.7.7 Temperature Sensor**

The temperature sensor used should provide an electrical signal that can be read by the controller. Parameters considered when determining the sensor type are linearity, range, sensitivity, and size. Table 12 below summarizes the differences between different sensor types.

*Table 12 -* Sensor type Comparison



The two main parameters that need to be observed are sensitivity and linearity. Size is not as important because all sensor types meet the project’s needs for this parameter. Thermistors have high sensitivity but their resistance decreases nonlinearly as temperature increases. The relationship between temperature and resistance is modeled by the Steinhart-Hart equation. Thermistors are great for sensing the temperature of a boiler because they achieve stabilities greater than 0.005 C. RTD sensors are not as accurate as thermistors since they are only suitable for 0.05 C stabilities. In addition, they have non-linear properties. The linear integrated circuit sensors are not desirable since their maximum allowable temperature is too close to the temperatures used in this application. In addition, they have a maximum error of 4 C.

**4.7.8 Op-amp Parameters**

There are many different characteristics examined when selecting an op-amp for a design. Many times there are trade-offs that need to be examined depending on the application. Table 13 below examines the different op-amp parameters.

*Table 13 -* Op-amp Parameters and their definitions

|  |  |
| --- | --- |
| Parameter | Definition |
| Input Offset Voltage | the voltage that must be applied between the two input terminals for an output of 0 V. This parameter can drift depending on temperature. |
| Input Offset Current | the difference between the currents at the input terminals when the output is zero. |
| CMRR | The ability of a differential amplifier to reject the signals common to both inputs. It is described as the ratio of differential voltage gain to common mode voltage gain. |
| PSRR | The ability of an op-amp to reject an AC signal riding on a nominal input DC voltage. |
| Gain Bandwidth Product | The ability of an op-amp to reject an AC signal riding on a nominal input DC voltage. |
| Output Current and Voltage | A higher output current allows for a larger load. Since the load amplifiers used for the PID design will not be used to drive the boiler, staying within the maximum allowed limits will not be a challenge |
| Settling time | the length of time for the output voltage to approach and remain within a certain tolerance of its final value. |
| Slew Rate | the rate of change of an op-amp’s voltage output over time when the gain is 1. |
| Phase Margin | The difference between the phase of the response and -180° when the gain is 1. |
| Harmonic Distortion | Occurs when an op-amp introduces harmonic frequencies into a signal and the output is distorted. This is sometimes grouped with noise (THD+N) to describe most sources of distortion. This is one parameter that needs to be observed for to achieve an accurate response. |

These are several types of noise in op-amps worth discussing when selecting an op-amp:

-shot noise: Noise pertaining to the potential barrier from the discreteness of electric charge. It is not dependent in temperature, just the current flow.  
-thermal noise: It occurs in passive resistive elements because charge carriers vibrate depending on temperature. It is also dependent on bandwidth and resistance.  
-flicker noise: Occurs in all active devices and is the result of the direction flow of direct current.

**4.7.9 Op-amp Comparison**

The amplifier used should be a single supply amplifier for convenient PCB integration. It should also have a high slew rate for optimal response times and very low noise for little distortion. It should also have a relatively high operating temperature in case the boiler is radiated across the machine. The amplifiers compared are the OPA 725, OPA 365, NE 5532, and TL084.

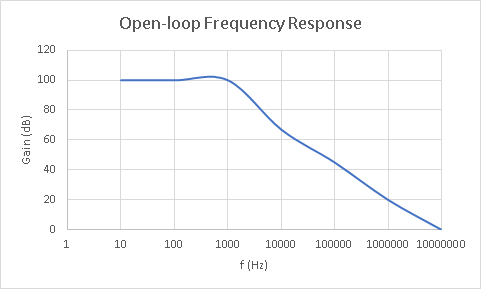
OPA 725: This is a single-supply op-amp usable with a supply from 4 V up to 12 V. It also features a convenient rail-to-rail output with an error of just 150mV. Its main features are great AC performance with low bias current, excellent CMRR (120 dB), PSRR, and high open loop gain and gain bandwidth product. They are also stable in unity-gain configuration. All of these parameters make it a good op-amp to use for process control application.

OPA 365: This is another single-supply op-amp with low distortion. It also features high CMRR and high precision with a low settling time. It also features a greater gain bandwidth product at a lower noise voltage but sacrifices in the slew rate. It costs a lot less than the previous option.

NE 5532: A dual supply op-amp that promises a lower noise and faster slew rate. Even though it is dual supply, it can be ran as a single op-amp if needed if configured correct. It is also priced lower in comparison to the other two because it is a general-purpose op-amp.

TL084: Another general-purpose amplifier similar to the NE 5532. However, it does not promise the required low noise that the NE5532 and the other low-noise operational amplifiers offer. In addition, its CMRR is the worst among the ones being compared.

The op-amp chosen is the NE5332 from Texas Instruments. Even though the first option, a phase margin of 60° as well as a faster slew rate and settling time, the OPA 365 has a higher CMRR and much lower offset voltage and noise compared to the OPA 725. These characteristics are more important for process control applications. It is also much cheaper than the other reasonable choice. However, using low noise amplifiers would not allow for the high output voltage needed. The TL084 does provide this but has a very high maximum offset voltage, low CMRR, low gain bandwidth product, and very high noise voltage. Even though the NE 5532 clearly falls behind in other characteristics such as slew rate, it is a good general-purpose operational amplifier with surprisingly very low noise and high output drive capability. This is essential to be able to control the solid-state relay. It also features a max offset voltage and maximum output swing bandwidths. Refer to table 14 below for a comparison between low-noise op-amps. Data for Figure 27 below was gathered from the NE5332 data sheet.



*Figure 27 -* Open-loop frequency response

*Table 14 -* Low-Noise Op-amp Comparison

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Product Number | Gain Bandwidth Product (MHz) | Phase Margin (°) | Slew Rate  (V/µS) | CMRR (dB) | Max Offset Voltage @ 25 C (mV) | Noise Voltage  (nV/√Hz) at 100kHz | Settling Time, 0.1% (ns) | Price (USD) |
| OPA 725 | 20 | 60 | 30 | 94 | 1.2 | 6 | 350 | 1.25 |
| OPA 365 | 50 | 45 | 25 | 120 | 0.1 | 4.5 | 200 | 0.80 |
| NE 5532 | 10 | 45 | 9 | 100 | 0.5 | 5 | Not specified | 0.31 |
| TL084 | 4 | 45 | 13 | 86 | 3 | 18 | Not specified | 0.80 |

**4.7.10 Tuning Procedure for PID Loop**

Tuning of the PID loop can easily be done through computer simulation with a transfer function model of the output. In this case, the best way to tune the PID loop is to test it manually. First, bring the process to normal operation with an output of 25% to 75%. Set all gains to 0 and then increase the proportional gain until the control loop gets stable with an offset from the setpoint. It is adjusted so there is a slight overshoot but a well damped response before settling out at steady-state. If the output keeps oscillating, increase the proportional gain. Then set the integral time to 1.5 times the time between the first overshoot and first undershoot of the step response. Re-adjust the proportional gain if needed. Set the derivative to one fourth of the integral term. Ideally, this should keep temperature constant even if sudden disturbances are sensed in the output. Another method of tuning a PID loop is by using the Ziegler-Nichols method. However, this will not be used in this project since because the alternative is much simpler and yields accurate results.

**4.7.11 Troubleshooting Common PID Problems**

A correctly tuned PID controller can still have unexpected results. The following is a list of common problems and how to go around to troubleshoot them.

1. Cycling: Load temperature varies around set point.

If the temperature sensor is not attached adequately to the load there will be a thermal time delay. This is because the thermal delay causes the controller to responds to a delayed error signal. This same problem will occur if the current limit is low. To fix this, increase the integrator time constant.

2. Poor Temperature stability: load may be exposed to large variations in ambient temperature. Isolate load if needed.

3. Noise on load: Use filter capacitors to the temperature sensor and use shielded cables.

4. Integrator Wind-up: If the control variable reaches the actuator’s limits, the system suffers from integrator wind-up. When this is the case, the controller becomes open-loop since there is no source of feedback. This causes build-up of the integral term, excessive output response, and large transients. One way to fix this is to choose a different integral term to constrain the process output within feasible bounds.

**4.8 Boiler**

**4.8.1 Heating Elements**

Heating elements convert electricity into heat. They achieve this through resistive heating. A good heating element has a high melting point, high resistivity, it should not oxidize or corrode, and a low temperature coefficient to keep resistance constant at high temperatures. The main materials used are nichrome, kanthal, cupronickel, and platinum. There are two main quantified characteristics of a heating element: its voltage and the wattage. All of the following heating methods benefit from a PID controller.

Large Single Boiler: This refers to a do-it-yourself water heater with one or more immersion heaters (Folded tubular heating element). Most of these elements will work only when submerged in water. Otherwise, they might burn out. A low density water heater element will slow sediment buildup, resulting in more durability. The large single boiler is first filled to about half capacity. When hot water is needed, the boiler increases the stored water to the desired temperature. A PID controller keeps temperature stable constantly. After extraction is complete, water temperature is increased to 255 F to enter steaming mode. Machine then cools back down to brew temperature. This boiler has the benefit that it can heat and hold hot water so it’s always ready for use. A different temperature can be used for steaming milk. However, it does not allow for brewing and steaming at the same time. Depending on the size of the boiler, it can take around ten minutes when water is first heated. Since water is being heated constantly, this type of boiler uses a lot of energy.

Small Single Boiler: Refers to boilers already used in coffee machines that can easily be purchased through a seller of machine parts. Works similarly to the large single boiler but only heats up small amounts of water when needed. This means it uses significantly less energy since it only operates when coffee needs to brewed. Since it does not need to heat up as much water, reaching brewing temperature happens much quicker. Another benefit of this design is that temperature is controlled a lot better because they come in an enclosed form factor. This design is more practical for smaller coffee machines that do not need to serve many milk-based drinks.

Double Boiler: separate boilers are used for simultaneous steaming and brewing. This also allows for different temperature settings between the two processes. After use, each boiler’s temperature remains stable and can be used immediately for the next serving of coffee. Its main disadvantage is that it uses a lot more energy than any other type of boiler. They are also more expensive and have a more complex design that requires more maintenance. This type of boiler is for someone who wants the best temperature stability for both steaming and brewing at a premium price.

Heat Exchange: a heat exchanger brews cold water directly from the reservoir via a copper tube designed to heat the passing water to the optimum brewing temperature. This means fresher water due to a lack of mineral deposits from a boiler. The boiler simply needs to keep the water at steaming temperature (255 F). The main disadvantage of this type of boiler is that the two temperatures are not independent of each other. This would bring into play the thermal characteristics of the copper heat exchanger along with complications in the PID controller implementation. In addition, running the machine back-to-back would also result in imprecise brew temperatures. In the other hand, this process does allow for simultaneous brewing and steaming without the disadvantages of a double boiler. Hence, a heat exchange boiler has lower maintenance, lower price, and lower energy consumption than the double boiler because it only uses one boiler. This is a great option for serving milk-based drinks on a regular basis if high temperature brewing accuracy is not needed.

Thermoblock: a thermoblock is a different type of water heating element that flash-heats water as it passes through an internal winding channel. It is made of two brass or aluminum blocks welded together. The fact that it is not always on means a decreased energy consumption. For steam, higher temperature is used and small bursts of water is sent through. Its main disadvantage is the unstable temperature due to a low thermal mass. This leads to undesired ‘wet’ steam. In addition, it does not allow for simultaneous brewing and steaming. They are also prone to blockages due to the narrow pathways. Another disadvantage is its durability; a thermoblock only last 3-5 years because they are very prone to leakage.

**4.9 Refrigeration Unit**

In this project, our group will build a small refrigeration unit from scratch, in order to be able to store milk and dispense it to drinks of users who request it. The refrigeration unit will be stored inside our coffee machine, and be directly integrated with the other internal systems of our product. Few coffee machines on the market offer this kind of functionality, which will help to differentiate our coffee maker from others on the market.

Our group’s initial consideration on how to build a compact refrigeration unit was vapor-compression refrigeration. This method is one of the most widely used method for air-conditioning of buildings and automobiles. This process involves circulating a liquid refrigerant inside the area to be cooled, and then dumping that excess heat outside. There are four main components to a vapor-compression refrigeration system: a compressor, a condenser, an expansion valve, and an evaporator. When compressor is started, it draws the low-pressure vapor from the evaporator and compresses it isentropically, increasing its temperature. It is then passed to the condenser where it is cooled, rejects heat, and gets converted into liquid. The liquid is then passed through the expansion valve which converts it back into vapor of low dryness. Then low temperature refrigerant passes through the evaporator where it absorbs latent heat from the heat chamber and gets converted into vapor. The diagram below illustrates the vapor-compression refrigeration system described above.

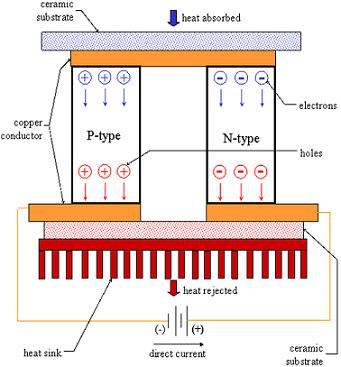
Upon further research, we decided that this type of refrigeration system would not be ideal for our application. There is no effective way in which we could house the entire vapor-compression system inside our coffee machine, as well as having to purchase and store the refrigerant, which would add to maintenance costs of the machine. Therefore, our group decided to explore other more viable options.

Our refrigeration unit will feature a thermoelectric cooling module as the main processing power. Thermoelectric coolers work by the Peltier effect. The effect creates a temperature difference by heat transfer. By applying a voltage across the conductors and creating an electrical current. When this current flow through the junctions of the conductors, heat will be removed from one junction and the excess heat will be deposited in the other. A heat sink will be attached to the hot side to maintain it in ambient temperature and avoid any possible hazardous conditions inside the coffee machine, while the cool side will house the milk. Thermoelectric coolers consist of p-type and n-type semiconductors heavily doped with electrical carriers.

Two considerations were made by our group during research on how to go about acquiring a thermoelectric cooler:

1. Buying the Peltier module, heatsink, and fans separately.
2. Buying the entire pre-assembled unit.

After searching the internet on pre-assembled thermoelectric coolers, our group found the pre-built units cost an average of $20. Therefore, we decided to purchase the unit to focus on building the rest of the housing for the refrigeration unit. Refer to the figure 28 below for a schematic on a thermoelectric Peltier cooler.



*Figure* *28 -* Thermoelectric Cooling module

**5 Design**

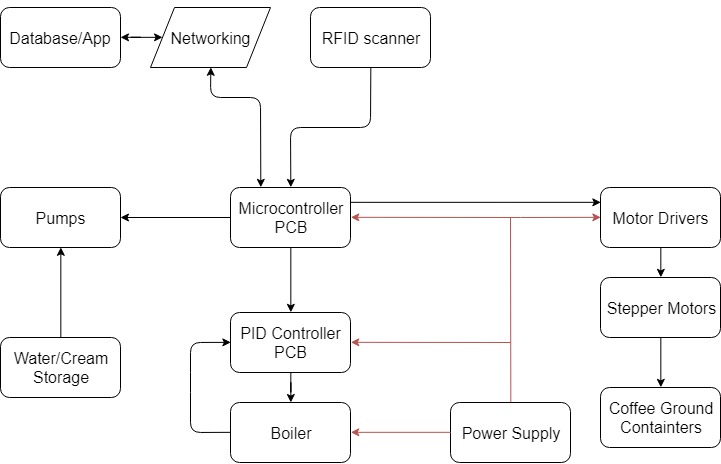
The sections below will go into detail about the design aspects of all the subsystems of the system. The sections will be broken up into hardware and software designs.

**5.1 Hardware Design**

The following sections will describe how we designed the hardware aspect of the Moka coffee machine and how all the system’s main components were put together to create the functioning smart coffee machine.

**5.1.1 Hardware Block Diagram**

The block diagram shown in Figure 29, is a representation of the top-level architecture of our system. All of Moka’s subsystems are centered around the microcontrollers embedded in the PCB. The microcontrolle PCB then connects to all of the rest of peripherals including the pumps, RFID reader and motor driver. The power supply will directly power only the boiler, the two PCB and the stepper motors.

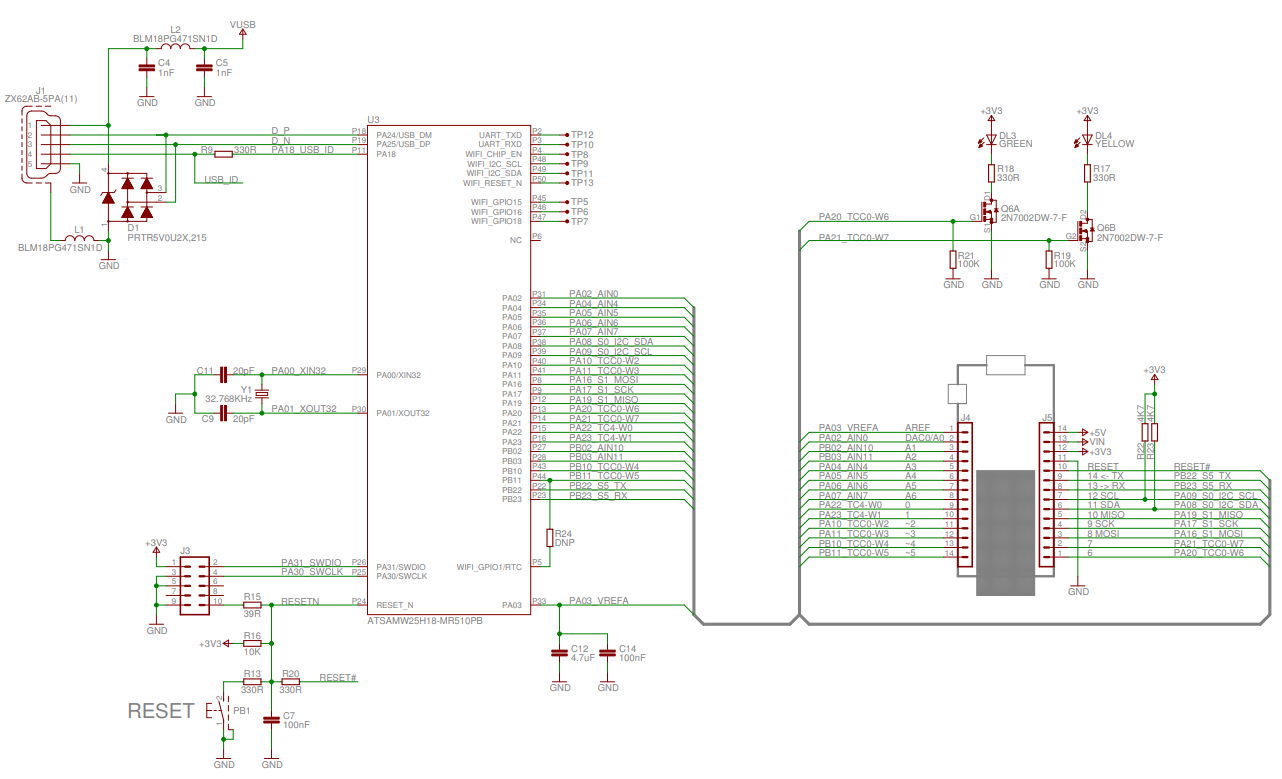


*Figure 29 -. Hardware Block Diagram*

**5.1.2 Microcontroller**

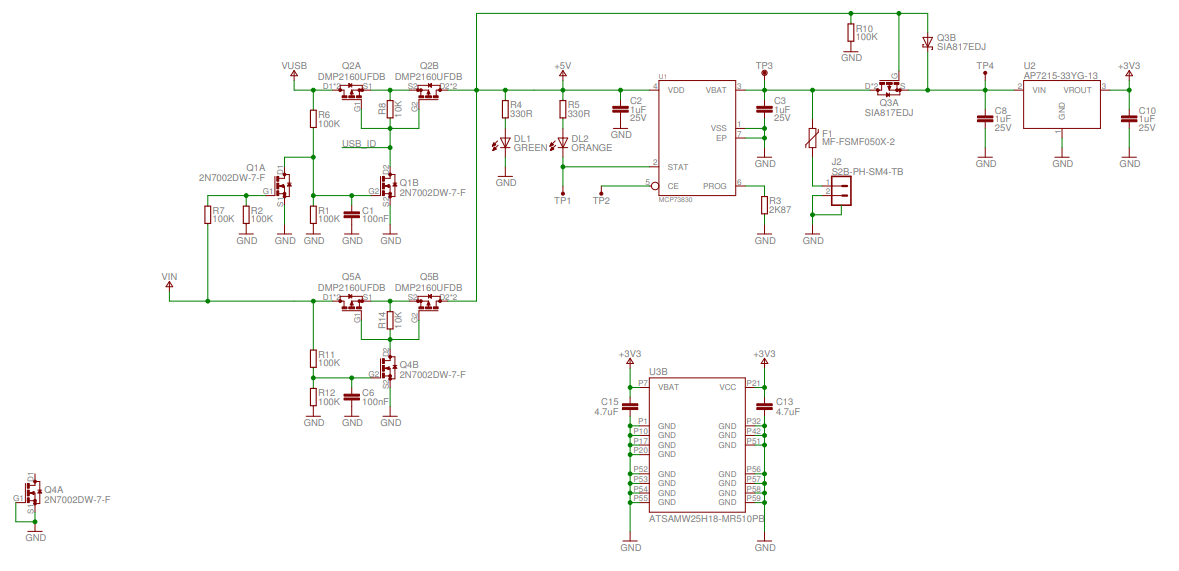
This section of the report will detail the procedure that was used to generate the schematic diagrams for the microcontroller. The software used to generate the schematics was Eagle. Eagle is a computer-aided design (CAD) software which facilitates the design of schematics and turning them into PCBs.

The microcontroller chosen for our project was the Arduino MKR1000. Arduino provides schematics of all its boards to facilitate users seeking to implement them into their own PCBs. As our application did not require a custom designed PCB for the microcontroller, we have used the schematics provided by Arduino in their website in our design. Figures 30 and 31 below show the schematics for the Arduino MKR1000 that were obtained from their website.

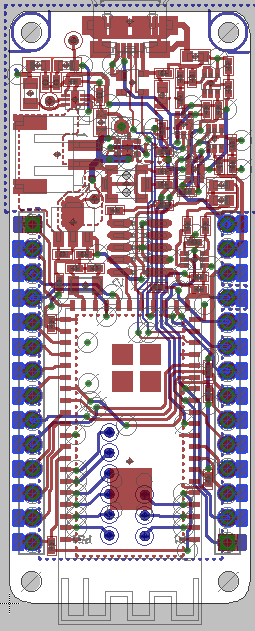


*Figure 30 - Arduino MKR1000 Schematic (1/2)*

After loading the schematics file in the software, we are able to use Eagle to generate the board we will be using in our project. The board layout file is also provided by Arduino, and so we will be using implementing their design. Figure 32 below shows the final design of the board we will be using in our coffee machine.



*Figure 31 - Arduino MKR1000 Schematic (2/2)*



*Figure 32 -* *Arduino MKR1000 Board Layout*

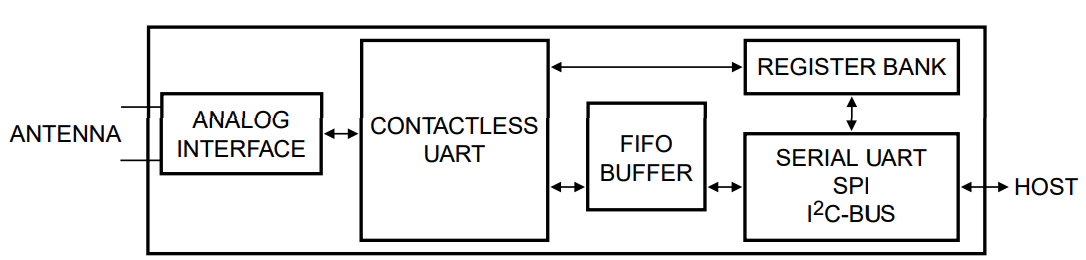
Senior Design II:

During senior design II many changes were made to the design involving our microcontroller unit. We tended to lean towards a more customized version of the board for our system. We stripped out the ATSAMW25 integrated microcontroller and placed it on a customized printed circuit board that would have simple plug ins for all of the peripherals in our system. The schematics for all of the final PCB designs can be seen in section 6 of this paper.

**5.1.3 RFID Module**

The main interface that connects the user to his/her settings in the database is the RFID scanner built into the Moka coffee machine. The scanner scans the unique 64 bit ID present in each RFID sticker. That ID is to be used to make a request to the server which returns the users coffee preferences and execute a payment transaction.

The first RFID module that we came across was the MFRC522 RFID Reader/Writer. The MFRC522 is a cheap RFID module that can read and write 13.56MHz RFID tags. The MFRC522 operates over the SPI serial communication interface which consists of a single master node, our main microcontroller, and multiple slave devices, one of which being the MFRC522. Figure 33 is the block diagram for the MFRC522 showcasing serial communication interface.



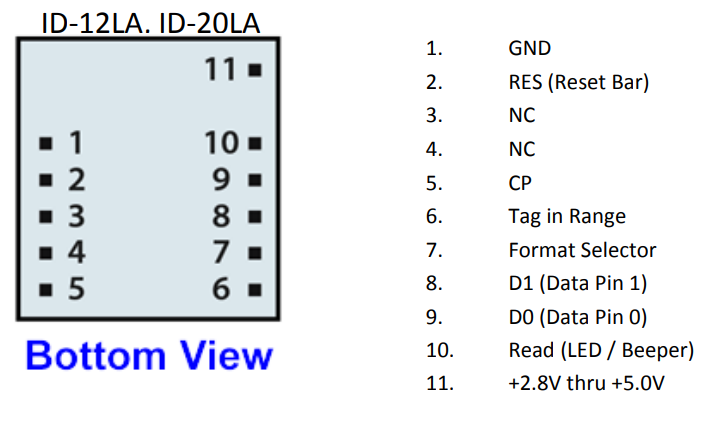
*Figure 33 -* MFRC522 Block Diagram

The problem we had with the MFRC522 is that the SPI serial communication interface is meant to be used with many compatible devices. Since the only SPI compatible device in our coffee machine is the MFRC522, the amount of pins required to connect the SPI protocol on the microcontroller is too much. Not counting the VCC and ground pins, the MFRC522 requires five pins to operate. Considering our microcontroller is smaller and has less pins than the Arduino Uno this RFID scanner module is not recommended.

The next RFID modules that we found were the ID-12LA and the ID-20LA. These RFID modules were manufactured by ID-Industries and utilizes a UART serial communication interface to transmit the RFID unique ID information to the main microcontroller. The ID-12LA and the ID-20LA are identical except the ID-20LA has a larger antenna with a longer range. Since we don’t need the longer range on the scanner, we are using the ID-12LA.

The ID-12LA fits the needs of the Moka machine perfectly. Although the MFRC255 is significantly cheaper, due to the  ID-12LA’s use of UART as the serial interfacing technology instead of SPI like the MFRC255, the ID-12LA only requires one data pin as opposed to the five pins required by the MFRC255. This along with the easy to use libraries available make the ID-12LA the optimal choice for the Moka coffee machine.

To configure the ID-12LA to work for our needs, we need to pull some pins to VCC and some pins ground. Figure 34 shows the pinout for the ID-12LA and the ID-20LA.



*Figure 34 -* ID-12LA and ID-20LA Pinout

To configure the ID-12LA for the Moka machines we have to configure the pins according to table 15.

Once the ID-12LA is configured, we need to implement the code that will read the unique ID off of the RFID sticker. To do this, we will utilize Thumperrr’s Arduino ID-20 Library for the Arduino IDE. This library is compatible with both the ID-20LA and the ID-12LA. Since our microcontroller will be running Arduino code, this library is ideal for us to use. This library allows us to simply wait for the ID-12LA to receive a signal from the RFID tag and then it will simply read the ASCII byte content to a string contained in memory. With this ID we can make all the requests necessary to the backend to facilitate the coffee transaction.

*Table 15 -* Moka RFID pin connections

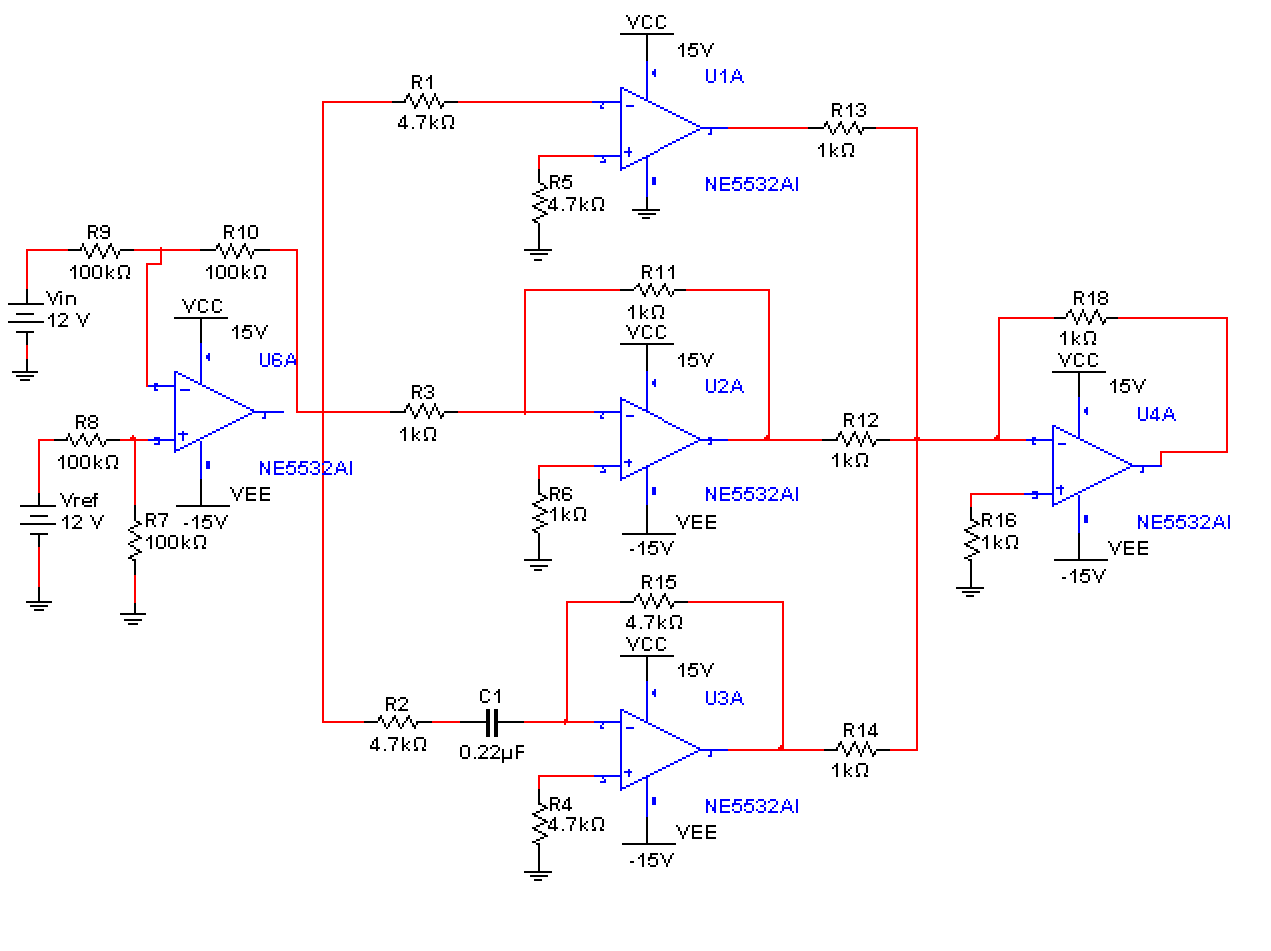
|  |  |
| --- | --- |
| Pin Number | Configuration |
| 1 | GND |
| 2 | +5V |
| 3 | Not used |
| 4 | Not used |
| 5 | Not used |
| 6 | Not used |
| 7 | GND |
| 8 | Not used |
| 9 | Data pin attached to microprocessor |
| 10 | LED to indicate successful RFID read |
| 11 | +5V |

Senior Design II:

During the test trials done during the construction of our system we determined that all the above RFID modules were not preferential for use within our PCB. The module we chose for our system was the PN532 NFC NXP RFID Module V3. This module allowed us to communicated with our microcontroller PCB over I2C using the SDA and SCL pins available on our microcontroller. This was a preferential change for us due to the available libraries that allowed the RFID tag ID to be read in a desirable formatted and allowed the use of less pins on the microcontroller.

**5.1.4 PID Controller**

The original values that will be used in the design are seen below in figures 35-40. The values for each gain term of the PID terms will be modified to obtain the desired gain during the tuning procedure. This requires testing with the actual load that will be used. The right-most op-amp’s output will be made into a pulse width modulated signal to control the solid state relay. This is the same way modern temperature controllers regulate temperature digitally.



*Figure 35 -* Op-amp PID schematic with difference amplifier input

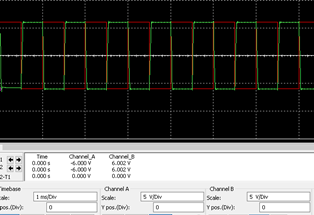
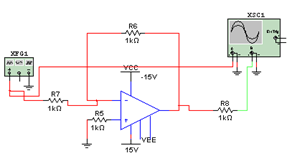


Figure 36 - Proportional Term Simulation

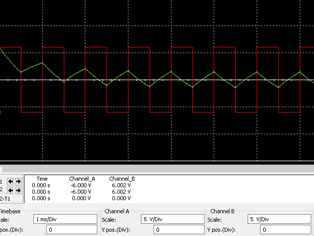
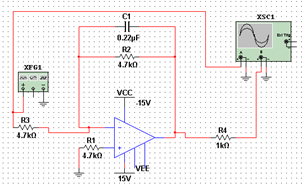


Figure 37 - Integral Term Simulation

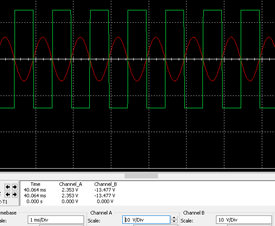
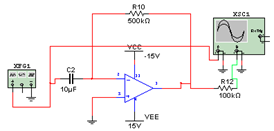


Figure 38 - Derivative Term Simulation

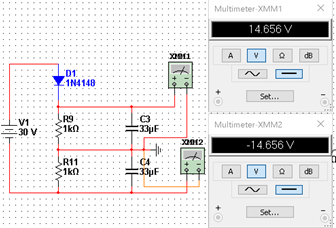


Figure 39 - Circuit used to provide power to op-amps

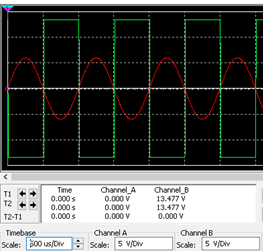


Figure 40. Op-amp multivibrator used to make output into a PWM signal

**5.1.5 PID Box**

A box will be used to house the PID controller and the parts involved such as the solid state relay. This will facilitate testing and protect the temperature-dependent electronics. The box used will be an 8” x 8” x 4” junction box.

The box will be a plastic electrical box as seen in the parts list. Externally, an LCD display will show the current temperature of the sensor for easier testing. It will also have a small LED light to show that there is power. The solid state relay will be a 24-380 V AC from Uxcell that comes with a heat sink. A hole in the box will be cut in order to accommodate the solid state relay and heat sink. The heat sink will need to be attached to the solid state relay and be kept outside of the box for heat dissipation purposes. Figure 41 below shows a similar PID box design for a digital PID controller. Tuning the temperature controller externally will not be possible once the box is closed unless digital potentiometers are used. A DK4N din rail will be used to help with wiring all of the parts together. These are suitable for this application since they are meant for power distribution; they can handle up to 30 amps and 600 volts. The specific temperature sensor used will also have to be wired to the PCB inside the box. Assorted parts such a screws, wires, and connectors will be used as needed.

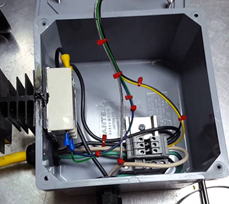


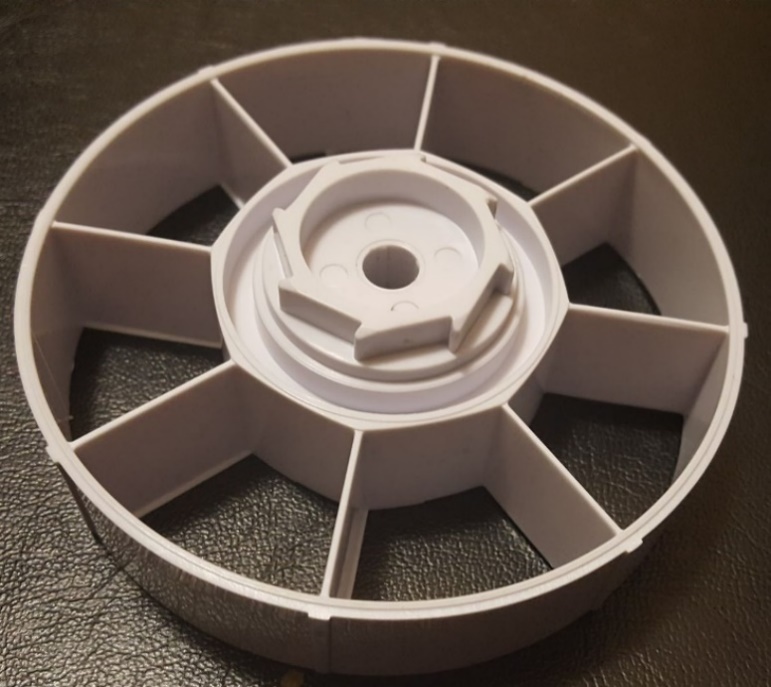
Figure 41 - Wiring of Solid State Relay with dinkle controller inside PID box

**5.1.6 Coffee Grinder/Dispenser**

This subsystem of Moka’s design is responsible for dispensing the coffee grounds that will be used to brew the users cup of coffee. There will be one of these designs for each available type of coffee that Moka will be available to brew. For our prototype design we will only be dealing with two different flavors of coffee, but the amount could be expanded based on the results of our initial prototype. There are three main components to this dispenser’s design:

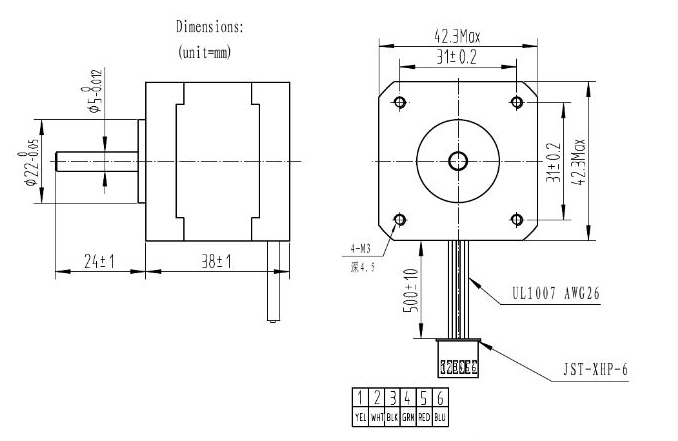
* Zevro KCH-06079 Indispensable 1/2-Pound-Capacity Coffee Dispenser
* 12Soyo SY42STH38-0406B Unipolar Stepper Motor
* A4988 Stepper Motor Driver Carrier

The container that will store the coffee grounds while the machine is inactive is a repurposed Zevro KCH-06079 Indispensable 1/2-Pound-Capacity Coffee Dispenser. The Zevro’s base is a plastic container that can hold up to a half a pound of coffee grounds. It has an airtight lid that will keep the coffee grounds fresh as well as make refilling the machine an easy process. The Zevro dispenser was originally designed to be a mechanical coffee dispenser that measures out one tablespoon of coffee grounds per each squeeze of the handle. After disassembling the device, it was discovered that with each pull of the trigger the device would turn a wheel that holds the coffee grounds to be dispensed. The wheel, shown in figure 42, can hold up to 8 tablespoons of grounds. With our design we look to automate this process by attaching the stepper to the dispenser wheel.



*Figure 42 - Ground Dispenser*

The stepper motor that this system is utilizing in specific is the Soyo SY42STH38-0406B Unipolar Stepper Motor. This motor is a professional high precision 12V stepper motor. The motor has a resolution of 1.8 degrees/step and a torque of 36 oz./inch. The motor has a step angle accuracy of ±5% on a full step when there is no load attached to it. If half step or micro stepping modes are used, even higher precision can be achieved. The precision of the stepper motor is the main reason a stepper motor was chosen over a normal servo. These precise steps will allow the dispenser to release the same amount of grounds for each customer permitting the motor drive has been setup correctly. A normal DC servo when used for continuous motion will eventually lose its timing and with this system that would lead to an inaccurate amount of grounds being dispensed. The motor is relatively small and light weight, as shown in Figure 43, and will be easily attached to the wheel within our ground container. With a resolution of 1.8 degrees/step the motor will need to move 25 steps to dispense one tablespoon of coffee grounds. In senior design II we determined that the best amount of grounds to dispense would be two tablespoons instead of one to produce a more potent cup of coffee for the user. This change was made with a simple alteration of the system’s code.



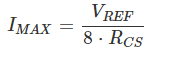
*Figure 43 - Stepper Motor Dimensions - reprinted with permission from RobotShop*

The stepper motors will then be attached to a stepper motor driver which will control and power the motors. This system design will be using a A4988 stepper motor driver carrier. This motor driver operates from 8V to 35V and features current limiting, over-temperature protection, and can allow micro stepping down to 1/16 of a step. The different step resolutions are based on the selector inputs MS1 MS2, and MS3. In Table 16 below shows the selector input combinations for each of the five possible resolutions for this motor controller. This specific controller is listed as a bipolar motor controller but can be used with our 12V unipolar stepper motor as well. Moka’s main microcontroller will be connected to this driver and will be what sets the specific step and directions values for the motor controller so our system will operate as designed.

*Table 16 - A4988 Resolution Chart - Reuse with permission from Pololu Robotics*

|  |  |  |  |
| --- | --- | --- | --- |
| MS1 | MS2 | MS3 | Microstep Resolutions |
| Low | Low | Low | Full Step |
| High | Low | Low | Half Step |
| Low | High | Low | Quarter Step |
| High | High | Low | Eighth Step |
| High | High | High | Sixteenth Step |

To get the best performance out of our motors as possible we will be using the highest practical voltage we can to power the motors. However, safety precautions must be taken so that any voltage spike does not go above the motor’s rated voltage range. This can be done by using the A4988’s built in trimmer potentiometer for active current limiting. One way for us to set the maximum current would be to calculate the reference voltage (VREF) based on our desired current. Shown in Figure 44, is the formula provided by Pololu Robotics and Electronics that will be used to calculate our desired VREF. RCS in this equation is the current sense resistance which 50mΩ for our version of the driver.

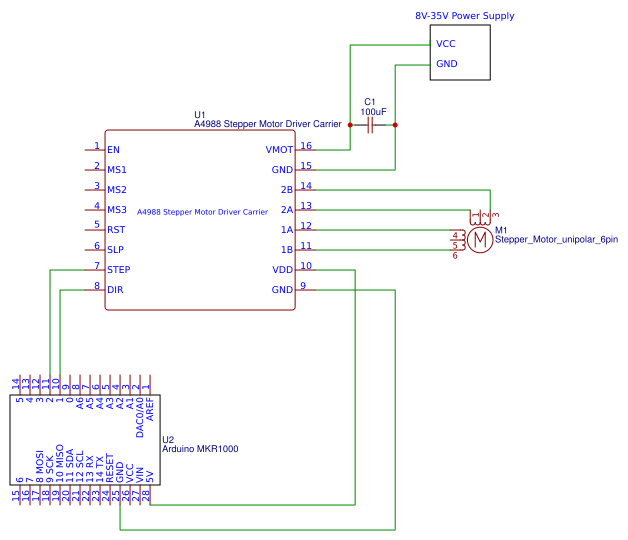


*Figure 44 - Formula used for Current Limiting - Reprinted with permission from Pololu Robotics*

The wiring diagram for this coffee ground dispenser subsystem can be seen in Figure 45 below. For full step each motor driver will require two I/O pins, which could potentially be knocked down to only one because there will be no need to change the direction of the motor once the initial firmware has been setup. These two pins will be attached the STEP and DIR ports on the motor controller. Then the 5V and GND pins will be connected in order to power the board. GND and VMOT on the motor controller are connected to a separate power supply of 8 to 35 Volts that will be used to power the motor. Here we will also use a decoupling capacitor with 100 µF in order to protect the controller from any possible voltage spikes from the power source. The last thing to be connected is the motor itself. Since we are using a unipolar motor with the A4988 Bipolar motor controller the wires must be configured correctly in order for the motor to be powered. Referring back to our motor schematic above in Figure 43, the center leads 2 and 6 will stay disconnected from the motor controller because they will not be needed to any bipolar operations in our system. This will leave the remaining wire lead pairs, 1/3 and 4/6 to be connected to 2A/B and 1A/B respectively.

Senior Design II:

One of the main things we decided initially was to use the highest possible voltage of 35 V to make our stepper motors as efficient as possible. However with our main power supply being 12V and with no other peripheral requiring more than 12V, we decided to use 12V to power our stepper motors. This was not a concern for our design and the machine was still able to operate in a efficient state.

*Figure 45 - Dispenser Subsystem Schematic.*

**5.1.7 Boiler**

The boiler that will be used is a single boiler with a resistance assembly since it provides the most benefits for the coffee machine application. The boiler that will be used is a 1300 watt 120 volt boiler. This particular model is used in many commercial coffee machines. There are no extensive datasheets on any of the boiler options since these are mostly replacement parts. Hence, a lot of experimentation will have to be made. This boiler was chosen because of its easy access to wire the resistive element, allowing for convenient wiring. The boiler has pre-installed sensors that are used to prevent the machine from overheating. A thermocouple will be attached next to the temperature sensors. Multiple thermocouple measurements will be made to determine the best placement of the thermocouple. Even if the temperature the thermocouple is not the one the water brews at, having it close to this setup will allow the controller to either increase or decrease heat accordingly. The sensor reading will need to be calibrated for accurate readings.

The temperature sensor of choice is the a K-type Thermocouple because of its wide range in temperature reading, lower price, and because it is the standard for similar applications. K-type thermocouples are the best choice for this application because it is a. As mentioned in the research section, an RTD sensor is not used since it is typically twice as expensive, has slower measurement speed and a smaller range of temperatures. The specific thermocouple that will be used is the Bayonet Style Thermocouple with stainless steel cable from Omega. This specific thermocouple can handle temperatures up to 480 °C as well as multiple options for mounting arrangements. Its durable and flexible stainless-steel cable is another reason why it was chosen over other thermocouples. If reading temperature by the microcontroller is desired, a module such as the MAX 6675 can be used with a secondary thermocouple.

A solid-state relay will be used to regulate the AC power provided to the boiler. The PWM signal that will control the solid-state relay will be provided by the output of the PID controller. The solid-state relay will have a heat sink attached to dissipate power. The solid-state relay that will be used is a normally open AC to AC 40 amp 24 V-380 V solid state relay from Uxcell as shown in the parts list. The wiring diagram can be seen in figure 46 below. This is a zero-crossing solid state relay since operation starts whenever the AC input voltage is close to 0 volts. Any voltage within the 3 to 32 volts is capable of switching the relay on. This is useful since the output range will not be known until the PID controller is assembled. This wide range of operation ensures that the relay will work for this application.

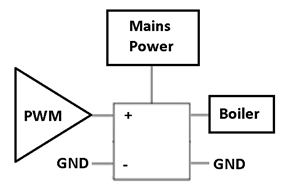


Figure 46 - Wiring of Solid State Relay to control boiler

Senior Design II:

One this we discovered with our design in senior design II was the fact that we chose the wrong boiler for our system. Instead of using a single boiler we used a one that was meant for an espresso machine which operates with extreme pressure. This became the only major issue with our system because it the water used to brew our coffee not be heated to the optimal temperature. This problem could be overcome by the replacement of our boiler with one that would be preferential for drip coffee and then the system would operate as initially intended.

**5.1.8 Water Tank**

Water was stored by a small tank inside the coffee machine. The tank inside the machine will be a 5-gallon plastic container. In final version of the system modifications to this tank will be made to get water into the tank through a water reservoir. The amount of water in the tank will not have to be precisely controlled. There are many ways to control the tank through simple circuits. Since this is not a main component of our project, a simple float switch as shown in the parts list will be used. This type of switch allows water into the tank if a floating object is below a certain level and stops water going in after it reaches a maximum point. The height at which the tank starts filling up again will be more than the minimum amount of water required to fill one large coffee cup and clean the machine. The maximum height will be close to full capacity of the tank.

Senior Design II:

For the systems initial demonstration, a simple one gallon water supply was used to supply water to our system. With this current design the water would have to be changed out quite frequently. But in future iterations of the design the system will be connected to a water supply to remove this constraint.

**5.1.9 Refrigeration Unit**

Senior Design II:

This whole section can be disregarded in terms of the final design of the system. Due to time and monetary constraints the refrigeration unit was not implemented into the first prototype. In future prototypes and the final iteration of the system, a refrigeration unit will be implemented to give more customization to the system’s users. For now, non-perishable creamer was used instead of milk.

Senior Design 1:

The goal behind the coffee machine is to deliver a simple yet convenient product to its users. Therefore, our machine features a built-in refrigeration unit which is used to refrigerate milk, and add it to certain drinks based on whether the user wants it or not. The milk is kept inside milk cartons and is delivered to the users drink via a peristaltic pump. The following section will detail the parts used and the design procedure that will be followed to implement the refrigeration unit inside our coffee machine.

**Casing** The exterior of the refrigerator is made of Poly(methyl methacrylate), commonly known as acrylic or acrylic sheets. It is used in a wide range of applications such as instrument clusters for vehicles and appliances. Acrylic sheets have several benefits that will be greatly beneficial to our project. It is more cost efficient and more resistance to impact than glass, as well as having great electrical resistivity. It is relatively inexpensive and widely available, making it an ideal choice for the refrigeration. Acrylic glass is transparent and durable, which will facilitate the handling and transporting of the milk, which will need to be exchanged regularly, whether the milk carton is empty or is past its expiration date. Since our refrigerator will house a gallon of milk, the first consideration we came across was which size to make our unit. We began by measuring the size of a standard gallon of milk, which came out to be approximately 6”x6”x10” (length x width x height). However, in order to comfortably house the electronics in the back of the unit, we will need to increase the size of the refrigerator. By increasing the size to 7”x12”x11”, we should be able to comfortably store either milk gallons or cartons, as well as house the electronics which will power this project.

A key component to a refrigerator unit is how well it manages to insulate heat. If a refrigerator unit has poor insulation, it will not be able to efficiently keep the interior cool enough to perform its job, wasting massive amounts of energy in the process. A poorly insulated refrigerator unit will need to turn on and off way more frequently than it needs to. This will wear out the power supply which powers the unit, it will waste a tremendous amount of energy, which in turn will cause the coffee maker to use more electricity than it needs to, lowering its sustainability and manufacturability. Therefore, choosing which kind of insulation to use for our refrigerator was of utmost importance while proceeding with our design. There are many types of insulation on the market. Considerations were made for the following kinds:

* Fiberglass
* Mineral wool
* Cellulose
* Polyurethane Foam
* Polystyrene

After some research into it, our group decided to proceed with polystyrene foam board insulation. It is efficient, inexpensive, and widely available, and most importantly, it is extremely easy to work with. The insulation will be cut to fit and then be hot glued to the interior of our acrylic casing. Our goal with this is to increase the efficiency of our cooler module and increase our energy savings.

**Cooler** The heart of our refrigeration unit is a thermoelectric cooler. Successfully integrating it into our design will be vital to ensure our product is a success. As previously discussed in the research section, thermoelectric coolers are an inexpensive and easy way to refrigerate small areas, making it the ideal choice for our refrigeration unit. The cooler will be fixed in the back panel of our refrigerator. A square will be cut into one of the acrylic panels in order to fit the cooler in. The cutout will be made according to the dimensions of the cooler which are 16.1 x 9.8 x 9.6 cm. The insulation will help ensure that the thermoelectric cooler is able to efficiently switch on and off based on our temperature sensor as needed. Since the two fans are heavy, we will also hot glue it to the surface in order to make sure it doesn’t disconnect from the mounting. Many factors went into choosing this specific cooler for our unit. The first factor was the ease of integrating it into our system. Since this unit already has the Peltier module attached into the heat sinks, it becomes easier to deal with than having to buy the two separately and manually fixing them together. A pre-built unit will reduce any errors that might come from integrating the components separately and will be more reliable in the long-term for our product. The working current of the thermoelectric cooler is 6A, which will require a more potent power supply. The cooler also operates at a voltage of 12 volts, which works nicely with the power supply chosen for the design. Figure 47 below shows the thermoelectric cooler we will be using in our refrigeration unit.



*Figure 47 -*Electronic Semiconductor Refrigerator Radiator Cooler

**Temperature Control** In order to efficiently control when the refrigerator turns on and off, our refrigeration unit will feature temperature control. This is necessary because without a system to constantly monitor the temperature, the refrigerator would need to be turned on all the time. This will wear out the power supply and result in immense energy losses and therefore adversely the overall performance of our product. The temperature control in our refrigeration will be performed by a digital temperature controller board. The controller comes equipped with a waterproof sensor and a one-channel relay, which will be connected to the thermoelectric cooler. The controller will be used to keep the refrigerator between 2 and 5 degrees Celsius, which is the optimal milk serving temperature. Whenever the sensor detects the temperature has fallen either below (although unlikely) or above that range, the relay will send a signal to the cooler and cause it to turn on until the inside of the refrigerator goes back inside the designated temperature range. The digital controller has a temperature range of -50 to 110 degrees Celsius, which will be more than enough for our application. With good thermal insulation the hot side of the thermoelectric cooler should never reach above 60 degrees Celsius, and the temperature should never dip below 2 degrees Celsius. We estimate that the refrigerator will only need to turn on every few minutes, making it an extremely efficient purchase for your design. The digital temperature controller operates at the same voltage as the thermoelectric cooler at 12 volts, greatly facilitating our choice of a power supply for our refrigeration unit as regulating the voltage coming into either of the systems will not be necessary. This was a great factor in our decision to include this controller in our design. Although the input current necessary to run the temperature controller is extremely small, between 35mA and 65mA, the necessary current to operate the thermoelectric cooler is 6A. This is important to take into account when deciding which power supply to choose for our project. Other smaller considerations were also take into account when choosing this specific controller. It is relatively inexpensive costing around $8. Its dimensions are 48.5 x 40 mm, making it extremely easy to incorporate it with the rest of the electronics of the refrigerator. Figure 48 below shows the digital temperature controller that we will be using in our refrigerator.



*Figure 48 -* HiLetgo Digital Temperature Controller Board

**5.1.10 Design Casing**

The case for our coffee machine enclosure was designed based off the machine shown on figure 49. Our case differs from the one shown in many ways. Our case is made of wood instead of plastic and have textured vinyl. The case dimensions are 30 inches tall by 20 inches wide by 25 inches long.



*Figure 49 - Model coffee machine*

The system utilizes an LED strip to display the status of the machine to the current user. When the machine is inactive, the LED strip is turned off. When a user scans their cup, the LED strip turns green to indicate a successful RFID read. If there is ever any errors, the LED strip pulses red. While the coffee is brewing, the LED strip pulses purple and yellow. Once the coffee is finished brewing, the LED strip pulses green until the user removes their cup from the scanner

Senior Design II:

Final implementations of the system design can be seen below in Figure 50.



Figure 50: Final Implementation of Case

**5.1.11 Power Module**

How you are going to power the electronics should be a major point of consideration in any project. Choosing a power supply which does provide the correct voltage and amperage to your components, can not only cause them to behave erratically, but in a worst-case situation, it even has the potential to be dangerous. Therefore, choosing the ideal power supplies to power the different subsystems in our coffee machine was a very important task. The next section details the process behind choosing the different power supplies, how they connected to their corresponding accessories, and the specific power supplies chosen for each task.

**Refrigeration Unit** The components of the refrigeration unit which need to be powered are:

* Thermoelectric cooler
* Digital temperature controller
* Small peristaltic pump

Thankfully, each of these peripherals operate at a voltage of 12 volts, which greatly eased the burden of finding a compatible power supply. Therefore, the only consideration that needed to be made for the refrigeration unit, was to find a power supply that could deliver the required amperage to the components. For the refrigeration unit, the thermoelectric cooler requires a working current of at least 6 A in order to function properly. As a rule of thumb, it is generally ideal to choose a power supply that is able to supply between 25 to 50% more amps than your maximum load requires. With this in mind we decided to choose the SP150-12 as the power supply for the refrigeration unit.

**Coffee Dispenser** Similarly with the peripherals of the refrigeration unit, we must check the input voltages and required amperage of the components for the coffee dispenser. The A4988 Stepper Motor Driver Carrier operates between 8 V to 35 V and can deliver 2 A of rated current. The Unipolar Stepper Motor requires an input voltage of 12 V and a current of 0.4 A. Given these specifications, we utilizes the same power supply as the refrigeration unit, the SP150-12 seen in figure 15. This is extremely efficient as we are able to power many components in our machine with a single power supply. The figure below shows the SP150-12, and table 17 summarizes the technical specifications for the SP150-12.



*Figure 51 - SUPERNIGHT 12V 30A Switching Power Supply*

Table 17 - Technical Specifications of SP150-12

|  |  |
| --- | --- |
| Output Watts | 150 W |
| Output Volts | 12 V |
| Output Current | 12.5 A |
| Input Voltage Range | 90 - 264 VAC |
| Frequency Range | 47-63 Hz |
| Operating Temperature | -10 to 60°C |
| Size | 7.8 x 4 x 2 “ |

**Arduino PCB** Similarly to the refrigeration unit, we needed a way to provide the correct voltage and amperage to the PCB. Checking the technical specifications sheet of the Arduino MKR1000, we see that it requires an input voltage of 5 V and has a 20 mA requirement. With these requirements in mind, we have opted for the MK40S-5 power supply. It is a small, enclosed, exterior power, which provides a voltage of 5 V and and amperage of 8 A. This power supply is also used for the filter cleaning subsystem.

For Senior design II we opted to use a DC-DC step-down buck converted to step down the 12V to 5V instead of using a separate power supply to power all of the peripherals that need 5V.

**Filter Cleaning** As detailed in section 5.1.14, the filter cleaning is performed by a digital RC servo motor, which flips the coffee filter over in order to dispose of the used grounds. The operating voltage range of the servo motor is 4.8V - 6.0V. The running current at no load is 270±10mA. Taking these requirements into consideration, we can see that the MK40S-5 power supply is able to supply the servo motor with the correct voltage and amperage for it to perform correct. Figure 52 below shows the MK40S-5 power supply, and table 18 the technical specifications for the MK40S-5 power supply.

During Senior Design II we realized that implementing a reliable way to clean the filter would not be feasible due to our lack of mechanical engineering skills, and the fact that the microcontroller PCB did not have enough pins for a RC servo motor, we have decided to not implement the feature.



*Figure 52 - MK40S-5*

Table 18 - Technical Specifications of MK40S-5

|  |  |
| --- | --- |
| Output Watts | 40 W |
| Output Volts | 5 V |
| Output Current | 8 A |
| Input Voltage Range | 90 - 264 VAC |
| Frequency | 47 - 63Hz |
| Operating Temperature | 0 to 60°C |
| Size | 4.75 x 2.5 x 1.5 “ |

**5.1.12 Peristaltic Pump**

Precise measurement of ingredients is a very important process in the design of a coffee machine. If the amount of water used in the brewing process in not correspondent to the amount of grounds used, the final product is not be satisfactory and leaves our effectively render our product worthless. Being able to accurately measure and transport a precise amount of water to the coffee filter, and afterwards milk to a customer’s cup of coffee is required. Therefore, our coffee machine made use of two peristaltic pumps, which transport both of these liquids accurately and increase the efficiency of the coffee machine. The following section details why we chose to use a peristaltic pump, the specific pump we chose, and how it connects to machine.

A peristaltic pump works by moving a liquid along a hose through compression and decompression. It operates in a very similar way to our intestines. A rotor first passes along the tube, compressing it. As the rotor turns, the tube is compressed along its length, creating a vacuum and therefore forcing the liquid through the tube. The part of the tube is before the rotor is then relaxed, allowing more liquid to flow through. This constant process of compressing is relaxing is the main working principle of the peristaltic pump, and allows for precise amounts of liquid to be transported and delivered to where they need to be in our coffee machine. A very important consideration to make is that the liquid is completely contained within the tube and never comes across any moving parts of the pump. This is worthy of note given how important it is to have the water and milk used in our coffee machine to not come across any impurities along the brewing process. If either liquid become contaminated during the brewing the process, then the peristaltic pump would not be ideal for our application and a replacement would need to be found. Peristaltic pumps also have incredibly low maintenance needs, which aligns with the overall goal of our coffee machine. They are able to be easily cleaned, which furthers lowers the need for maintenance in our coffee machine.

There are many different kinds of peristaltic pumps available for purchase in the market. Peristaltic pumps have a wide range of commercial applications, such as dialysis machines, media dispensers, concrete pumps, and aquariums. Therefore, finding a pump appropriate for our application was the first step taken after deciding we wanted to include it in our design. After researching on the internet, and looking through several do-it-yourself (DIY) projects, we decided to use a simple dosing pump. Our design includes two different kinds of pumps, a small pump for the refrigerator, and a larger one with a larger flow rate for the water reservoir. The smaller pump is connected to the refrigerator, and delivers milk directly to the cup of coffee after it has finished brewing. The larger pump delivers a precise amount of water from the water reservoir to the boiler. The amount of water delivered is dependent on the input from the user when they choose the size of the cup of coffee that they want. Additional specifications for this pump includes an input voltage of 12 volts, making it easily compatible with the power supply for both the refrigeration unit and the digital temperature cooler. It runs at current of 80 mA, which our power is more than enough to handle. The pump is inexpensive and easy to work with, making it an ideal fit for our project. It has a maximum flow rate of 100 ml/min, which should be more than enough to dispense both milk and water. Figure 53 shows the two different kinds of peristaltic pumps we included in our design.



*Figure 53 - Small Peristaltic Pump (left) and Larger Peristaltic Pump (right)*

For ease of comparison, the technical specifications of both pumps have been summarized in the table 19 below.

*Table 19 -* Technical Specifications of small and large peristaltic pumps

|  |  |  |
| --- | --- | --- |
|  | Small Pump | Large Pump |
| Motor Voltage | 12V | 12V |
| Motor Current | 80mA | 0.5 - 1.4 A |
| Relative Humidity | 80% | 80% |
| Temperature Range | 0 - 40°C | 0 - 40°C |
| Flow Rate | 0 - 100 ml/min | 0 - 400 ml/min |

It is important to address why we felt the need to go for two different kinds of pumps in our design. Users of our coffee machine are able to choose between three different sizes of coffee. These sizes are 6, 8, and 10 fluid ounces. Approximating one fluid ounce to be 30 milliliters and converting these sizes, we reach 177 ml, 236 ml, and 300 ml respectively. If we were to use the smaller peristaltic pump, which has a maximum flow rate of 100 ml/min, it would take 3 minutes just to transport the water from the reservoir to the boiler if a user were to choose a coffee size of 10oz. Considering that 3 minutes is only the time is takes to transport the it, and that the boiler still needs to heat up the water and brew the coffee, it becomes clear that the small pump is simply not powerful enough for the task. After exploring different options, we found a more powerful pump which can handle 400ml/min. Again, using our coffee machine’s biggest cup of coffee as an example, the time it would take to transport the water from the reservoir to the boiler is cut down to 45 seconds. Although we would still prefer if this time was even shorter, more powerful peristaltic pumps become significantly more expensive, which adversely impact the manufacturing price of our coffee machine. After weighing the cost-benefit analysis of a more expensive but higher performance pump, we decided to go with the more economical choice.

In Senior Design II, due to difficulties acquiring the larger peristaltic pump we have decided to make use of a diaphragm pump instead. It works the same as the larger pump, however it is much more powerful with a much higher flow rate. It is rated at 12V, but we have stepped down this voltage to 5V in order to compensate for the sheer power of the pump.

**5.1.13 Filter Cleaning Design**

The final consideration we must make in regard to our hardware, is how we clean the coffee filter. Coffee grounds must be cleaned after each cup has been made to avoid unwanted residue in the coffees to come. Small household coffee makers use paper filters which are manually replaced by the user every time they wish to make a cup of coffee. Our goal with this project, however, is to create an automated system that performs this action automatically after each cup of coffee has finished brewing.

To accomplish this the first choice, we made was to replace the conventional paper filters with a permanent plastic one. There are plenty of plastic filters available on the market, so choosing one was not difficult. The only requirements are that the filter must be able to handle a maximum of 10 fluid ounces of water, which is our biggest cup of coffee. With these requirements in mind we decided to go with the HIC Plastic Coffee Filter, seen in figure 54 below. It is inexpensive, and can handle 12 oz. of water, which is more than enough for our application. Refer to the figure below for a picture of the filter.



*Figure 54 -* HIC Plastic Coffee Filter –Permission Pending

After choosing our which plastic filter to use, the next hurdle to get through is to design a system that automatically cleans the filter. After exploring our options are researching similar projects, we have decided to utilize a 180° servo motor to flip the plastic coffee filter upside down. While the filter is upside we shower it with a stream of hot water coming from the boiler to clear it out. After searching the internet for servo motor options, we have decided to use the SunFounder Metal Gear Digital RC Servo Motor. This motor is inexpensive and relatively simple to use, making it ideal for our application. The motor takes input from the microcontroller to flip the coffee filter upside down after each cup of coffee. Table 20 summarizes the technical specifications of motor. Refer to figure 55 below for the servo motor we will be using in our coffee machine.



*Figure 55 -* SunFounder Metal Gear Digital RC Servo Motor – Permission Pending

Table 20 - Technical Specifications of the SunFounder Servo Motor

|  |  |
| --- | --- |
| Operating Temperature Range | -10°C to 60°C |
| Operating Voltage Range | 4.8V to 6V |
| Running current (no load) | 270±10mA |
| Operating speed (no load) | 0.18 Sec/60° |

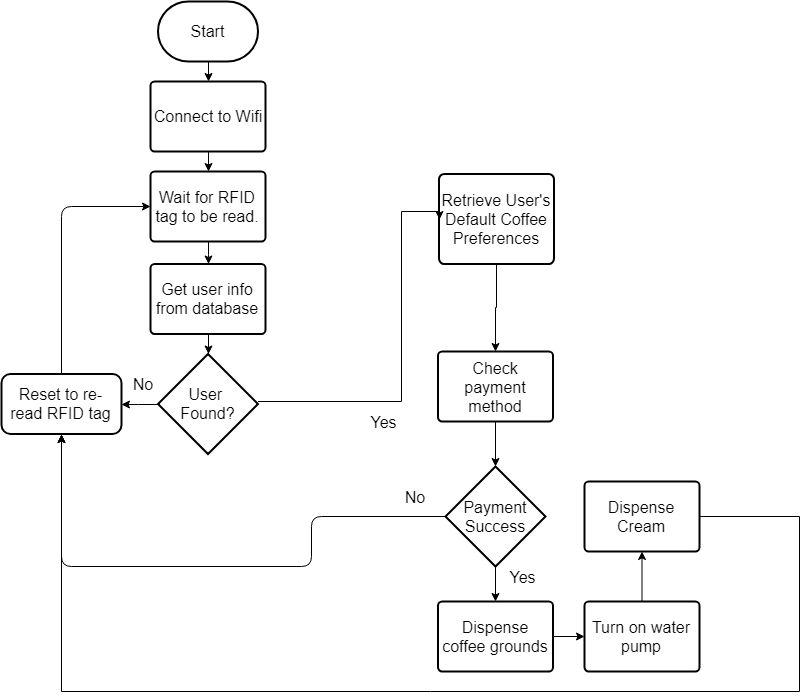
In Senior Design II, due to the technical limitations of our project, we have decided to not implement the servo motor to clean the filter. This feature will be implemented in the final revision of our product.

**5.2 Software Design**

In the following sections we will be going over the design choices that we made for the software aspect of our project.

**5.2.1 Software Block Diagram**

The Block diagram shown in Figure 56, shows the basic operation flow for the software in Moka’s system. Once an RFID has been read by the receiver, the program searches the database to make sure they are a user and that they have the correct credentials. Then the program goes on to check the queue of app data to find the corresponding drink for their order. The system waites until an order under the corresponding ID is found in the queue or manually selected from one of the switches on the machine. After a drink is confirmed, the user’s stored payment information is charged. On success of a payment, the machine then signals to all of the subsystems to start brewing their cup of coffee.



*Figure 56 -* Software Block Diagram

**5.2.2 Database**

The database handles the storage of all models being used by our system.  The models include the User Model and the Coffee Machine Model. The database is hosted on a server that can be used to transmit and receive messages to the network.

To store the data, we chose to use MongoDB. MongoDB is a NoSQL database. A NoSQL database is a non-relational database which utilizes a variety of data models including document, graph, key-value, and columnar. MongoDB uses documents with JSON formatted data to store information.

MongoDB utilizes on-disk journaling for logging. Journaling allows for MongoDB to quickly recover write operations that were written to the journal but not written to any data files. This kind of issue can occur if mongod (the MongoDB daemon) quits unexpectedly for any reason. It is important for us to have safeguards to prevent any potential loss of user data.

The database data directory is located at /data/db on our DigitalOcean VPS. We created a user called mongo on the server and give it permissions to the /data folder. We are storing the MongoDB logs in /data/logs. Here we can view all the database’s history along with any errors that might occur during operation. The mongodb daemon is to run on port 27017 on the VPS. This port is to be utilized by the backend to communicate with the database.

The backend utilizes the local filesystem on the server to handle file uploads. In the event of a file being uploaded to system, the backend takes the form data and upload the file to a folder called “uploads” in the root directory of the project. In the event of a file being uploaded to system, the backend takes the form data and upload the file to a folder called “uploads” in the root directory of the project. The uploads folder is then be available as a static resource to the frontend whenever a file is needed.

The mobile application sends information to the backend Node.js server and the server utilizes a library called Mongoose to interface with the MongoDB database. Mongoose provides us with several functions that allows us to pull data from the database and send it to the user when needed.

Table 20 and table 21 detail the User Schema and Machine Schema for the MongoDB database. The name column contains the name that we use to reference the field in the database. The description column contains a short description describing what the field is used for.

*Table 21 - User Schema*

|  |  |
| --- | --- |
| Field Name | Description |
| ID | Unique ID for each user |
| Name | User’s first and last name |
| Email | User’s email address. This is used by the user to log into the mobile application to make modifications to their coffee preferences. |
| Password | User’s password. This password isbe hashed using the bcrypt hash function. When a user logs in, the provided password is hashed using the same algorithm and then compared. If they match, the user is issued an authentication token. |
| Payment Information | Contains payment details for payment processing |
| Name On Card | Name of card holder |
| Card Holder Address | Billing address |
| Card Number | Credit card Number |
| Expiration Date | Credit card expiration date |
| Security Code | Security code on back of credit card |
| Cups | Contains information about each cup the user owns |
| RFID UID | Unique ID associated with the RFID Sticker on the cup |
| Cup Size | Cup capacity in ounces |
| Coffee Preferences | Contains coffee preferences for each coffee setting |
| Grind Type | Which flavor grinds the user wants |
| Cream Amount | Quantity of cream in ounces |
| Sugar Amount | Quantity of sugar in ounces |
| Tokens | An array of JWT tokens for use with authentication |

*Table 22 - Machine Schema*

|  |  |
| --- | --- |
| Field Name | Description |
| ID | Unique ID for each coffee machine |
| Location | Location of the coffee machine |
| Status | The status of the coffee machine (e.g. Idle, Running, Out of order) |
| Token | JWT token for coffee routes access |

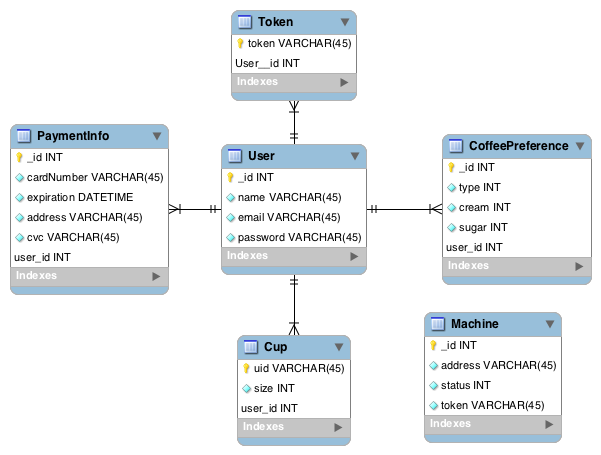
Table 23 and table 24 detail the required fields in the database for the user and machine model respectively. The field column contains the key that MongoDB uses to reference the data, the data type field contains the data type of the given field, the required column states whether or not the field is required for each instance of the user or machine. 57 shows the database model diagram.

*Table 23 - User Model Specifications*

|  |  |  |
| --- | --- | --- |
| Field | Data Type | Required |
| \_id | ObjectId | Y |
| name | String | Y |
| email | String | Y |
| password | String | Y |
| payment\_info | Array[Object] |  |
| payment\_info.name | String | Y |
| payment\_info.address | String | Y |
| payment\_info.cardNumber | String | Y |
| payment\_info.expiration | Date | Y |
| payment\_info.cvc | String | Y |
| cups | Array[Object] |  |
| cup.uid | String | Y |
| cup.size | Number | Y |
| coffee\_prefs | Array[Object] |  |
| coffee\_pref.type | Number | Y |
| coffee\_pref.cream | Number | Y |
| coffee\_pref.sugar | Number | Y |
| tokens | Array[String] | Y |

Table 24 - Machine Model Specifications

|  |  |  |
| --- | --- | --- |
| Field | Data Type | Required |
| \_id | ObjectId | Y |
| address | String | Y |
| status | Number | Y |
| token | String | Y |



*Figure 57 -* Database model diagram

MMAPv1 is MongoDB’s original storage engine based on memory mapped files. It excels at workloads with high volume inserts, reads, and in-place updates. All records are contiguously located on disk, and when a document becomes larger than the allocated record, MongoDB must allocate a new record. New allocations require MongoDB to move a document and update all indexes that refer to the document, which takes more time than in-place updates and leads to storage fragmentation.

MongoDB automatically uses all free memory on the machine as its cache. System resource monitors show that MongoDB uses a lot of memory, but its usage is dynamic. If another process suddenly needs half the server’s RAM, MongoDB yields cached memory to the other process.

**5.2.3 App Backend Development**

Our system architecture matches closely with the Client-Server model.  The server acts as the central database, where multiple users are allowed to login and access the system, as well as take requests from clients to send and receive data. The server handles the storage of all coffee settings as well as information on all the users.  Users are able to access all user functions (coffee settings, payment information, etc.) and be able to update or modify the data. Because of the needs of our system, and our current description, the Client-Server model represents our system the best.

We are hosting our backend server remotely on a VPS (Virtual Private Server) on DigitalOcean. The backend server is running Ubuntu 16.04 as its main operating system. Once we have a public IP address for our server, we will be able to make requests to it from anywhere in the world.

Ubuntu 16.04 Package Dependencies:

* mongodb-org@v3.6
* nodejs@v8.x
* npm@v5.8.0
* git@v2.4.7

Backend Server Dependencies:

* axios@v0.16.1
* bcryptjs@v2.4.3
* body-parser@v1.17.1
* express@4.15.2
* express-fileupload@v0.1.3
* jsonwebtoken@v7.4.0
* lodash@v4.17.4
* moment@v2.18.1
* mongodb@v2.2.30
* mongoose@v4.9.7
* nodemon@v1.13.3
* shortid@v2.2.8
* validator@v7.0.0

The backend was created using the Node.js. We be designed an Restful API that is accessible by our coffee machines via online requests. To handle the creation of the API, we used the Node.js library Express. Express allows us to define GET, POST, PATCH, and DELETE routes for our API that allows the user to modify the data in the database from the mobile application. Creating an API also allow other developers to interface with our data. The route names and descriptions are shown on tables 24, 25, and 26.

*Table 25 - Authentication Routes*

|  |  |  |
| --- | --- | --- |
| Request Type | Route | Description |
| POST | /login | Verifies user with the provided email and password. If the user is authenticated successfully, this route returns a JWT authentication token for the user to use for all future requests. If not, this route returns an error. |
| DELETE | /logout | Deletes the given JWT token from the tokens array in the user document. This route returns a successful message if successfully logged out or an error if unable to delete the token. |

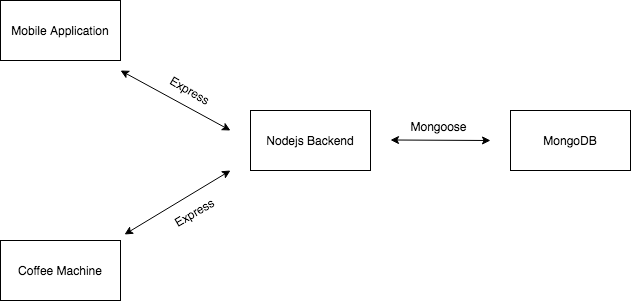
*Table 26 - User Routes*

|  |  |  |
| --- | --- | --- |
| Request Type | Route | Description |
| GET | /users | Returns a list of all users and all of their coffee information (useful for developers who want to take advantage of our API to gather statistical or analytical data). |
| POST | /users | Creates a user with the given data. The user data provided must adhere to the model detailed in the data models section. This route returns the new user as a response or an error if the data provided is invalid. |
| PATCH | /users/:id | Modifies the user with the given ID. This route is used by the mobile application to modify user payment information, coffee preferences and other information such as email or phone number. This route returns the modified user as a response or an error if the data provided is invalid. |
| DELETE | /users/:id | Deletes a user with the given ID. This route returns the deleted user as a response or an error if the user is not found. |

*Table 27 - Coffee Routes*

|  |  |  |
| --- | --- | --- |
| Request Type | Route | Description |
| GET | /coffee/:cupId | Returns the user’s default coffee preferences. This route is utilized by the coffee machine whenever the user puts their cup on the scanner. The server finds the user based on the cup ID provided. This route also triggers the payment processing API to execute a payment. If the payment is unsuccessful, an error is returned instead of the user’s coffee preferences |
| DELETE | /coffee/:cupId | Deletes the cup with the given cup ID from the list of cups for the user. |

To authenticate users, we utilized JWT tokens. A JWT token is a compact and self-contained way for securely transmitting information between parties as a JSON object. This information can be encrypted and decrypted by using a shared secret. We are encrypting user information and creating a token for every each user when they log in. These tokens are stored in an array per user document. When a user makes a request, their token is decoded and verified thus authenticating the user. By protecting every route, we protect all user information that we have stored in our database. The data flow model is showcased on figure 58.



*Figure 58 -* Data Flow Model

We also need to authenticate each individual coffee machine to prevent people from obtaining other users’ information. Each coffee machine is issued a JWT authentication token as well as users. These tokens are used to authenticate the coffee routes detailed above. Doing this makes the routes only accessible to users and the machines.

**5.2.4 App Frontend Development**

The frontend of our application is based around the React Redux Flow model. In this model, React is in charge of rendering views to the user and Redux is in charge of handling the data for the frontend. Figure 59 shows the React Redux Flow model.



*Figure 59 -* React Redux Data Flow

In the root folder we have two important files, package.json and webpack.conf.js. The package.json file is in charge of all the dependencies that are associated with our application and the webpack.conf.js configures Webpack to handle all the different loaders that are necessary for the app. Along with these two files, we also have the src folder. This folder contains four sub folders, the actions folder, the components folder, and the reducers folder.

During development, webpack is watching all of our Javascript and SCSS and compiling them into a single file, bundle.js and styles.css respectively. Those files along with index.html are the ones that are served to the user.

Frontend Application Dependencies:

* axios@v0.18.0
* bulma@v0.6.2
* file-loader@v1.1.11
* moment@v2.21.0
* react@v16.3.0
* react-big-calendar@v0.19.0
* react-dom@v16.3.0
* react-dropzone@v4.2.9
* react-redux@v5.0.7
* react-render-html@v0.6.0
* react-router@v4.2.0
* react-router-dom@v4.2.2
* redux@v3.7.2
* redux-form@v7.3.0
* redux-promise@v0.5.3
* redux-thunk@v2.2.0

Frontend Development Dependencies:

* babel-core@v6.26.0
* babel-loader@v7.1.4
* babel-plugin-transform-object-rest-spread@v6.26.0
* babel-preset-env@v1.6.1
* babel-preset-es2015@v6.24.1
* babel-preset-react@v6.24.1
* css-loader@v0.28.11
* extract-text-webpack-plugin@v3.0.2
* html-webpack-plugin@v3.1.0
* image-webpack-loader@v4.2.0
* node-sass@v4.8.3
* rimraf@v2.6.2
* sass-loader@v6.0.7
* serve@v6.5.3
* style-loader@v0.20.3
* url-loader@v1.0.1
* webpack@v3.0
* webpack-dev-server@v3.1.1

The frontend development dependencies are only utilized during development, meaning, once the application has been created, the dependencies listed in this section are no longer required and not included in the bundle.js

When the application first loads, React first renders the App component to the DOM encapsulated by a Redux Provider module. This provider module contains the current global state of the app which is handled by Redux. With the provider in place, the state can be accessed by any component via the mapStateToProps function.

The App component returns the login page if the user is not logged in. To check whether the user is logged in or not, the App component checks the isLogged in value of the auth object in the global state. If the user’s JWT authentication token is present in the local storage, the App component executes the fetchAll action which retrieves all the user’s information from the database, this includes user information and all coffee presets. The App component also handles the frontend routing for the application. As the url route changes, so do the components that are being loaded.

The data is loaded via the actions and then stored according to their given reducers. The action types are used to discern which reducer corresponds with which action payload. The actions and reducers are defined according to table 28.

*Table 28 - Actions and Reducers*

|  |  |  |
| --- | --- | --- |
| Action Type | Reducer | Description |
| FETCH\_ME | Users | Contains information about the current user, this includes coffee preference information and payment information. |
| AUTH\_LOGIN | Authentication | Sets the isLoggedIn value to true. |
| AUTH\_LOGOUT | Authentication | Sets the isLoggedIn value to false and deletes the saved JWT token from the local storage. |
| ADD\_COFFEE | Users | Adds a coffee preference to the user’s coffee preferences list. |
| DELETE\_COFFEE | Users | Removes coffee from the current user’s coffee preferences list. |
| EDIT\_COFFEE | Users | Modifies a coffee preference from the current user’s preferences list. |
| ADD\_CARD | Users | Adds a credit card to the user’s payment information. |
| DELETE\_CARD | Users | Deletes a credit card from the user’s payment information. |
| ADD\_CUP | Users | Adds a coffee cup to the current user’s list of coffee cups. |
| DELETE\_CUP | Users | Removes a coffee cup from the current users list of coffee cups. |

**5.2.5 User Interface**

Figure 60 is the login screen for the Moka coffee machine companion app. The first thing the user is presented with is the Moka logo. If the user doesn’t already have an account, they can click the button on the bottom of the page and they are prompted for a first name, last name, email, and password. Then they are directed to a page where they can create a new coffee preset. The machines use their default coffee selection any time they scan their cup at a machine. Once their account is created and they have entered payment information they are directed to the “coffee station” page. If the user doesn’t have an account, the user is directed to the create account page shown on figure 61.

|  |  |
| --- | --- |
| https://lh4.googleusercontent.com/JmRWwllwMjPjs3RH5d5ljycjhm5yEdyoy8XDdVFyVZ65xM3qBkOeRvBjsk-UoSdIhibzOr-usYmc1j6ARvHXhZbUGdbNkcNEwz0NXE9AC-E3PR5eJWCyRVG0I1tJswkkYHtFMfw- | https://lh4.googleusercontent.com/CaKjBN8FxF-9itUO3EC--_33Qh7jpVmH-H4Dl-fqkmF3l5l-VsaA0Bmy6Re83StJjabp256zzv4jMl6x5JOCmto7sLSb0YucUGoiGvzWK9XUeohkg4px7Qbz3BeGFuoJ_x0Kyf4O |
| *Figure 60 -* Application Login | *Figure 61 - Create Account* |

Figure 62 shows the coffee station page. This is home page for the application. Here the user has control over their coffee presets. They can choose which coffee preset is to be the default preset to be used by the coffee machine when they scan one of their cups. They can also chose to delete a coffee preset by clicking on the X icon on the top right corner of the panel. To add a new coffee preset, the user can tap on the “add” panel which is always be the last panel with the plus icon. This takes them to the “new coffee” page (Figure 64).

The navigation menu shown on figure 63 gives the user the ability to jump to different pages on the site. There are 4 main pages in the app. The “coffee station”, “cups”, “payment info”, and the “settings” page. The pages are to be showcased in the next few sections. In the navigation menu, the user also has the ability to sign out of their account.

|  |  |
| --- | --- |
| https://lh6.googleusercontent.com/J7vpwBSvyPdZgKtfCKuvdC0nDpOn4vt2HeEw6SVypzumNI64cuTQgVmHKxqLJq5gwUo4R227XpxujwYr0nMJQGb8frsRb9WlMHXY2dr6H_NXc6rSFKLm3CzQEGgu8SevQfTLmHAC | https://lh6.googleusercontent.com/jC7V0s36YCB--Dk05M5OfVT-eHekIKzXp9Inznjo_QunCvstJVXsOKQLogT8D-x8t8MrG3HgCLQYcMMCZY0ffkDujfaQLTHPcro9016NpnXjc1XIuN_mq2HRhPfLGIWU4u0HehjD |
| *Figure 62 -* Coffee station page | *Figure 63 -* App navigation menu |

Figures 64, 65, 66 showcase the coffee preset creation. This is the process that the user goes through when tapping on the “add” panel in the “coffee station” page. The user is first prompted to select their coffee flavor. This is based on the grind flavors that are available in the Moka coffee machines.

Once they have selected their desired flavor, the user then chooses how much cream/milk they want in their coffee. They can utilize the slider to choose a percentage of milk.

Next, they choose how much sugar they would like. The values here range from 0g to 5g of sugar. Once they tap on the finish button, they are taken back to the coffee station page and their new preset is available to choose from.

|  |  |  |
| --- | --- | --- |
| https://lh6.googleusercontent.com/0FJcpf87cv4CuUqLK0cVAIZR2AA_M7dhCPSoryg724Az4i7NjD1g8Z2Y2BWRLE1y_0wj5l1RWU36L0f9FoJvHL2WEbtVvT2BEguHNm5QqpdNUdXIU2XacIIlV1LxUQbN4BehE_nk | https://lh5.googleusercontent.com/w4Nh6_vdiOOR6EmhC9dVQ_M0-NIzPzhUigiDXawIVQO3GvUowh1cYTSBjdc10ew6q74Iqbq5xv6u9ERAcvo_RXPj3BQU8JVSQszd1QqnVT7oy2wrvn4aP_eyekttRvO9miFx9eta | https://lh3.googleusercontent.com/vWIfLRy58GV_GBRro3AUK135jyU8DYHPilYZbEMD4QQZc4eLOH-74pmMADJp4xgXeP1h98xDg5Zz3MRnO7Zf3jved3CEY5Ntwl3klxhJst69Y6YAGam7VzBslWnG_omxoKIGoPTF |
| *Figure 64 -* Flavor select | *Figure 65 -* Cream select | *Figure 66 -* Sugar select |

Figure 65 is the cups page. This page displays all the cups that the user has added to their account. These cups are associated with the RFID sticker that they have attached to their cups. They can dissociate a cup from their account by clicking the X button the appropriate cup panel. When one of the cups from this list is scanned at a Moka machine, the system authenticates the cup and the machine and send back the user’s default coffee preset. They can add a cup by tapping the “add” panel which are always be the last panel with the plus icon.

Figure 66 and figure 67 show the cup registration process. The first step in registering a user’s cup with their account is to scan the cups QR code sticker. Tapping the scan button brings up a QR code scanner for the user to scan their sticker with. Once scanned, the checkmark icon appears and the user can continue to the next step.

The final step for adding a cup, is to choose the amount of liquid the cup can hold which is shown on figure 69. The Moka machines use this metric to judge how much milk dispense and coffee to brew

|  |  |  |
| --- | --- | --- |
| https://lh3.googleusercontent.com/amnxBELY4uB-Wv80S45MhOL8zZmm6r5lk5WuNAGv13cumbKlV6h1Og7xrlYuORd_KJUca-VGWEBEWKIbZmUwwpjdKn9INbgOLxyIhyMKozKhHKdOYgEMek_Jl4Hsv9mO9DFMYUdI | https://lh6.googleusercontent.com/tPDKiFbFgoZj3zZuESzRmc6jFxWv0o3v-Ar8zVxNQcuCd-SPllZC4QNC-apjm3cUSrSB9INim1kjNIjXymUJMUVWU9RTAoFB9XbW8VwhpqhBmy39bC-7Nqj4Ki9FFqjGbUMCczYm | https://lh4.googleusercontent.com/l5e3eJdWwo2RWcL5NI-_mMC6-FJ-YTjalsPEYrz6yy5rZliBEJf7jNko-WjxRk5HdjFAPoVe87DAdYfvv54pOpFhgXEeOBtbSPqqQyXCY4467FDsZqXkM2HStw6Dys2HvwQxENsk |
| *Figure 67 -* Cups page | *Figure 68 - Scan page* | *Figure 69 -* Size select |

Figure 70 shows the payment info page. This page displays all the current user’s payment methods. Each credit card is displayed as a box below the large main area. In the main area, the user’s current default payment method is displayed. The default credit card is the card that is used when the user uses one of the Moka coffee machines. Before dispensing the coffee, the machine initiates a transaction on the credit card. If the transaction is successful, the coffee is dispensed.

Figure 71 shows the credit card addition process. To execute a transaction using the payment processing API, we need the card holder’s name, the card holder’s billing address, credit card number, expiration date and cvv (card verification value).

Senior Design II:

For senior design two the payment processing was not implemented for the purposed of the demo. However it is still implemented in the backend code should this s

|  |  |
| --- | --- |
| https://lh6.googleusercontent.com/ZmiIY8RlzKF4xMch8pCZiLiDrlojpDG_0AkC53Yhfu0FauxVMtGoBomSKpONSOK7beMkHLSr3Z-t724dG9PFyn9H3cQ2V8KMLENdhTcGKiNb9NmvRXKMPKIMbAhQLd3AfZkxo_AS | https://lh4.googleusercontent.com/ntnyJS-3GcftJazkslc3U1kgspwpHrrjO0BruP-isDTWwuH1HT7E9eskAX5eu3JKeW-1vvtAoaVEImlyAfrzV0IOIVmNMIDYrkD4RE8hTI7YDRRIWNScIpx_b0m3kNc_yFQg8bGq |
| *Figure 70 -* Payment info page | *Figure 71 -* Add payment info |

A user experience feature that is present in the user application is the usage of modal views to display important information. This can be seen in figures 72, 73, and 74. We use the modals specifically to get user confirmation when deleting a coffee preference, deleting a cup, and deleting a payment method.

|  |  |  |
| --- | --- | --- |
| https://lh5.googleusercontent.com/EcTCRaLlaR4SadNFTFNpnPQvluqMAlmFeH8O2a9457STuW8jOM-CEFG063D_MbJFDaNtFH-7roFyGgmgMgdtX05GGv1qZBUZHk6V_5jhGLqmDrE3DqgWWVg7J5bBdr1XQJLa4UWl | https://lh4.googleusercontent.com/gEIS8OUEPBAIAKoObzO4XR77PPNAxJU4cop2aB-ePwGMl-DIhSLCXeYVXLeqwpnPNNdO2d3gWE-51jx-x1QKuPCcPl5sj_AEhzYuxnzBdcUcDCR1r9ybc49Xg2geBXp3fEOiQnvr | https://lh3.googleusercontent.com/FLwQP3gTlM5OK32Zg8ZNnorIZKpf8bLmTfEiZWMNn7M_rLEioArZiA8LIZMZRxxK2QGMKVKugMtmVuHxRfC-eldAF3M0dSCOp5aNUkv3NSYZ6-pTQow7-HAQ8IX8ZUqy4PjjUYal |
| *Figure 72 -* Delete coffee | *Figure 73 - Delete cup* | *Figure 74 -* Delete card |

By utilizing these modals, we prevent the user from deleting any unwanted information.

**6 Prototyping**

The following section will outline some of the method to prototype the system as well as talk about some issues that could be run into along the way.

**6.1 Printed Circuit Board**

A printed circuit board electrically is where one connects components and connectors to each other. This is done through copper traces that allow signals and power to be driven to any desired point in the board. Layout of the PCB design is done through computer software. In addition to controlling where each component will be placed, the designer gets to control many other things such as the board dimensions and width of the copper traces. Research concerning PCB practices will assume that the software used is EagleCad. A PCB can be self-made using the wet-etch system. Since this is not a part of this project, no research will be made on the steps to producing a printed circuit board at home.

For testing a PCB, one should first inspect it visually for any obvious defects. After this step, there are two tests that can be done ensure the PCB is working as designed. The capacitance test will test for opens and shorts by sending a charge on the trace. One then probes each trace and measures the induced capacity. The second test is the resistance test. It is a way of measuring the resistance in ohms as the current flows through a trace.

**6.1.1 PCB Practices**

Although layout a PCB is very simple in principle, there are many things to watch out for to have a good design. The first one is to find a grid spacing that suits as many components in the design as possible. To reduce impedance and parasitic effects, it is also recommended to reduce the length of the copper traces as much as possible. Also try to group related components together since this makes finding errors much easier. For example, put the capacitor and resistors used in the integrator configuration of an op-amp close to the IC itself. To power ICs, use common rails for each supply. The trace width also needs to be taken into account depending on the amount of current that will be passing through the trace. Orient similar components the same way to reduce chances of mistakes during soldering. Also, do not place components on the solder side of the board in a way that interferes with other components. As for power and ground planes, make sure to have them internal and centered to the board. PCB software also has a design rule check feature that checks that many of the most important practices are being followed. Another reason why good PCB layout is also important is avoid heating issues. The first thing that needs to be done for this is to identify the components that dissipate the most heat and place the critical components far away from these. Adding thermal reliefs is also important to make the soldering process easy by avoiding heat sinking through the component plates.

Other practices to follow when designing PCB for op-amps is to make the connections to the inverting pin as short as possible. This is because long traces can act as antennas allowing high frequency noise to couple into the signal chain. For the inverting pin, long traces can cause stability issues. This practice also includes decoupling capacitors that go into the supply pins. Lastly, it is important to note to not sacrifice good layout for labeling on software.

**6.1.2 PCB Design**

There are many implementations of analog PID controllers with op-amps in parallel to represent the P, I, and D terms. After a lot of analysis, the implementation in the circuit below was chosen. The main reason for this is that it is very easy to follow, easy to test and easy to find gain terms. Other designs researched used different lag and lead op-amp configurations, used unfamiliar integrator or differentiator circuit configurations, or did not specify the recommended values for any parts of the circuit. As discussed in the PCB design section, the final circuit has several key differences such as a different operational amplifier. However, the basic principle and most components have stayed the same.

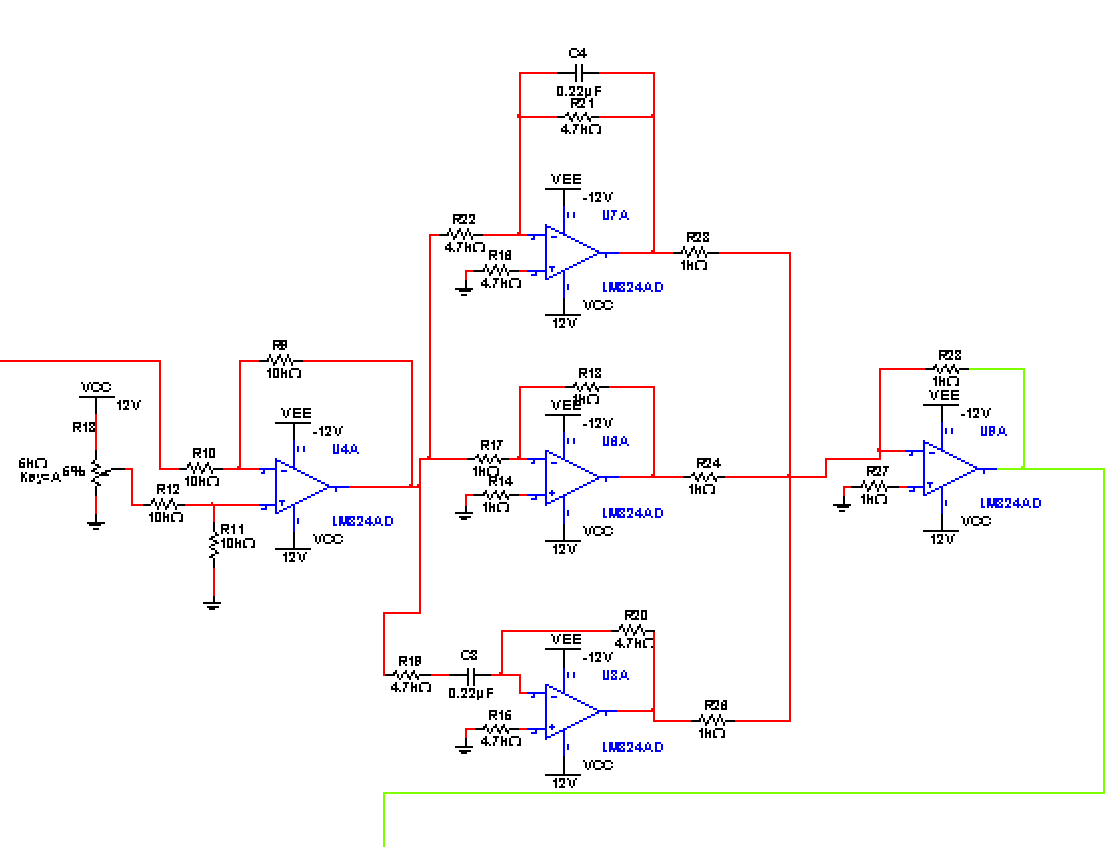
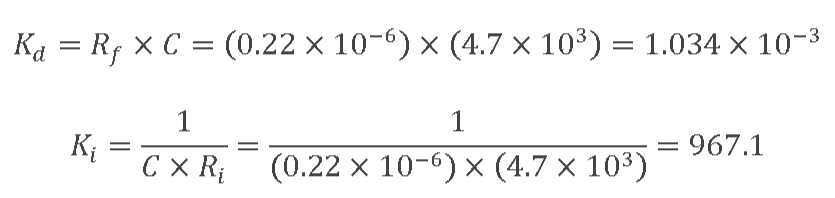
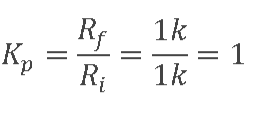


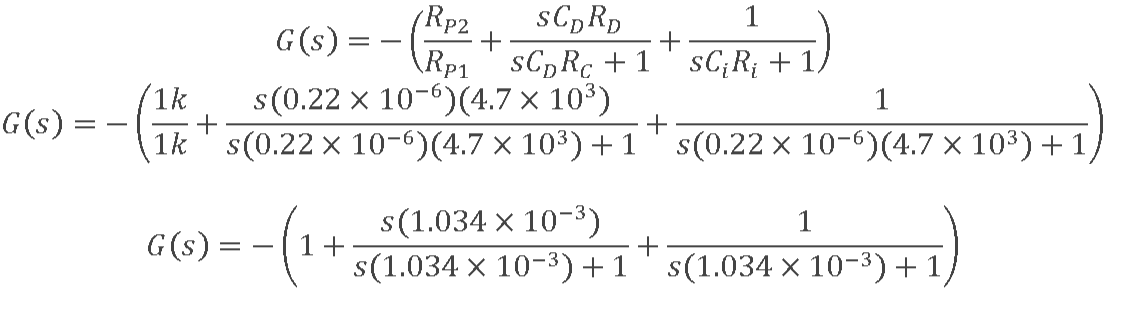
Figure 75 - Preliminary PID Circuit Design

To find the gain of the proportional, differential, and integral terms, the following formulas can be used. They are then solved for the values in the figure above. Since potentiometers will be used instead of fixed values, the gain of the P, I, D terms has to be recalculated after tuning the circuit.

C= 0.22µF, Ri = 4.7kΩ, Rf = 4.7kΩ



According to the table above, the transfer function for this circuit can be found using the following equation

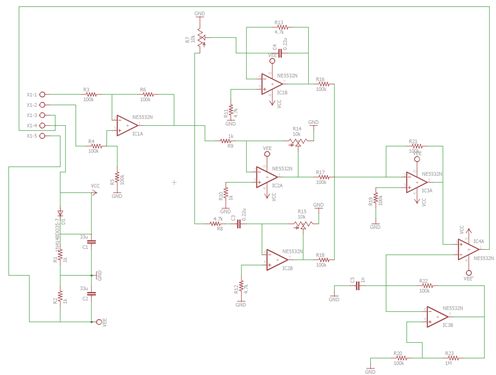


The table below shows all the components required for the PID controller design. Through hole components were used since small size is not a priority. No electrolytic capacitors are used since there are no high capacitances that need to be accounted for. For capacitors in the microfarad range, radial through-hole tantalum capacitors from Multicomp Passives will be used. The following characteristics of tantalum capacitors is why they were chosen: lower inductance, lower internal resistance, and improved decoupling. A radial film capacitor is used for the circuit attached to the inverting circuit of the last op-amp. It was chosen because of its small capacitance, lower internal ohmic losses and lower parasitic inductance. The diode chosen for the plus and minus voltage supply circuit is the general purpose 1N4148 as this meets the voltage, power and other application requirements. The NE 5532 used will be a standard dual in-line package from Texas Instruments. Each of these integrated circuit packages feature two operational amplifiers inside. An 8-pin dual in-line socket will be used to connect each operational amplifier into the PCB. Standard through-hole carbon film resistors will be used since high no high voltage or currents will be used. Resistors will have a 5% tolerance. Three 23-turn 10k trimmer potentiometers are used instead of the standard rotary ones to keep the size of the PCB relatively small. This will also keep the gains more constant since they need to be turned by a screw. The 23-turn will also make for very accurate resistor values. This is important to control gain with maximum accuracy. A terminal clock with for five connections will be used for connecting physical wires directly to the PCB.

Table 29 - List of all components required for PID PCB design

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Item | Quantity | Value | Package | Manufacturer | Price |
| Tantalum Capacitor | 2 | 33u | C2.5-3 | Multicomp Passives | 0.525 |
| Tantalum Capacitor | 2 | 0.22u | C2.5-3 | Multicomp Passives | 0.115 |
| Radial film capacitor | 1 | 1n | C2.5-3 | Vishay | 0.351 |
| Diode (1N4148) | 1 | 1N4148DO35-7 | DO35-7 | Multicomp Passives | 0.17 |
| NE 5532 op-amps | 4 | NE5532N | DIL08 | TEXAS INSTRUMENTS | 0.244 |
| Carbon Film Resistor | 4 | 1k | 0204V | FARNELL | 0.012 |
| Carbon Film Resistor | 11 | 100k | 0204V | Multicomp Passives | 0.549 |
| Trimmer potentiometer (T93YA) | 3 | 10k | B25U | VISHAY/SPRAGUE | 0.789 |
| Carbon Film Resistor | 4 | 4.7k | 0204V | Multicomp Passives | 0.25 |
| Carbon Film Resistor | 1 | 1M | 0204V | Multicomp Passives | 0.023 |
| Terminal Block | 1 |  | AK300/5 | PTR Messtechnik | 0.59 |

The figure below shows the design that was made using EAGLE 7.5.0. However, some key modifications to it were made for our application. Firstly, three resistors in the operational amplifiers that control the gains of the derivative, integral, and proportional terms were replaced with three trimmer potentiometers. This will allow for adjusting each gain term precisely. The second change that was made was using NE5532 operational amplifiers. Compared to op-amps recommended, these provide a lower noise and maximum offset voltage that are beneficial for the application. A circuit to control the polarity of the power was also incorporated. These will be used to power the operational amplifier (Vcc and -Vcc). A 120 V to 30 V step down transformer will be used for this configuration to make Vcc and -Vcc have magnitude of 15 volts. Another key change to the standard schematic is that a final operational amplifier will be used at the output. The input at its non-inverting terminal will be the output of the PID controller, while the input at the input terminal will be from an stable multivibrator. The combination of these two will result in a PWM signal that can be used to control a solid-state relay. The circuit will be adjusted as needed by modifying the gain of the summing amplifier.

Figure 76 - PID Design Schematic

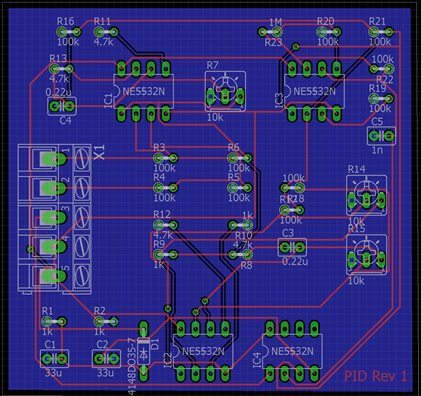
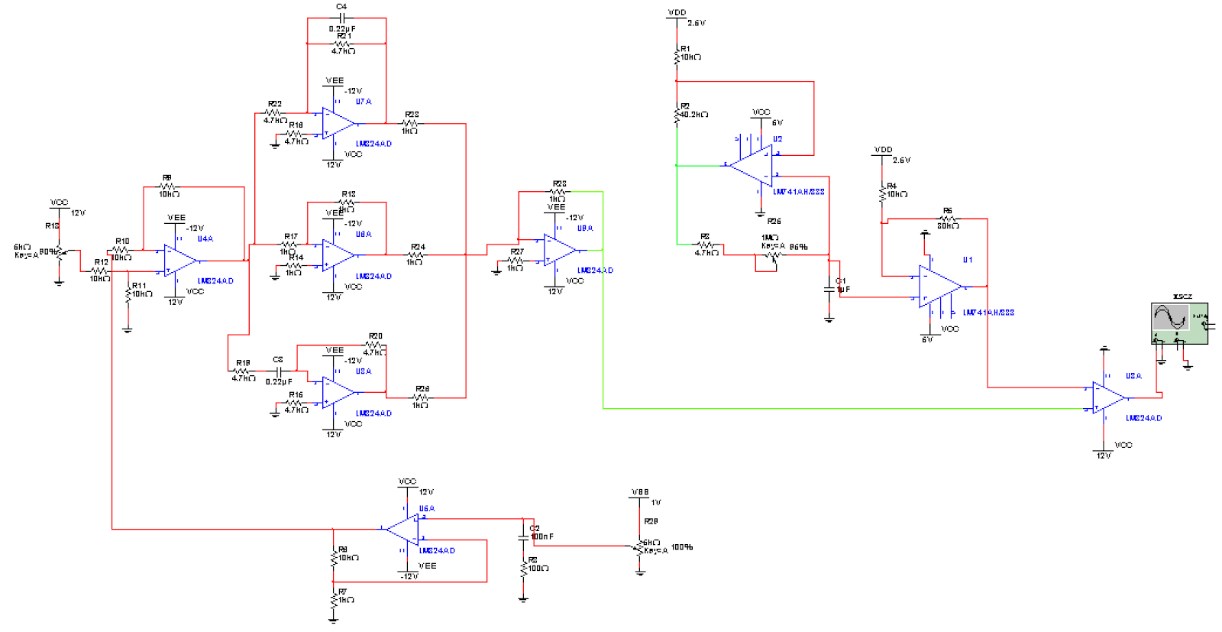


Figure 77 - PID PCB Design

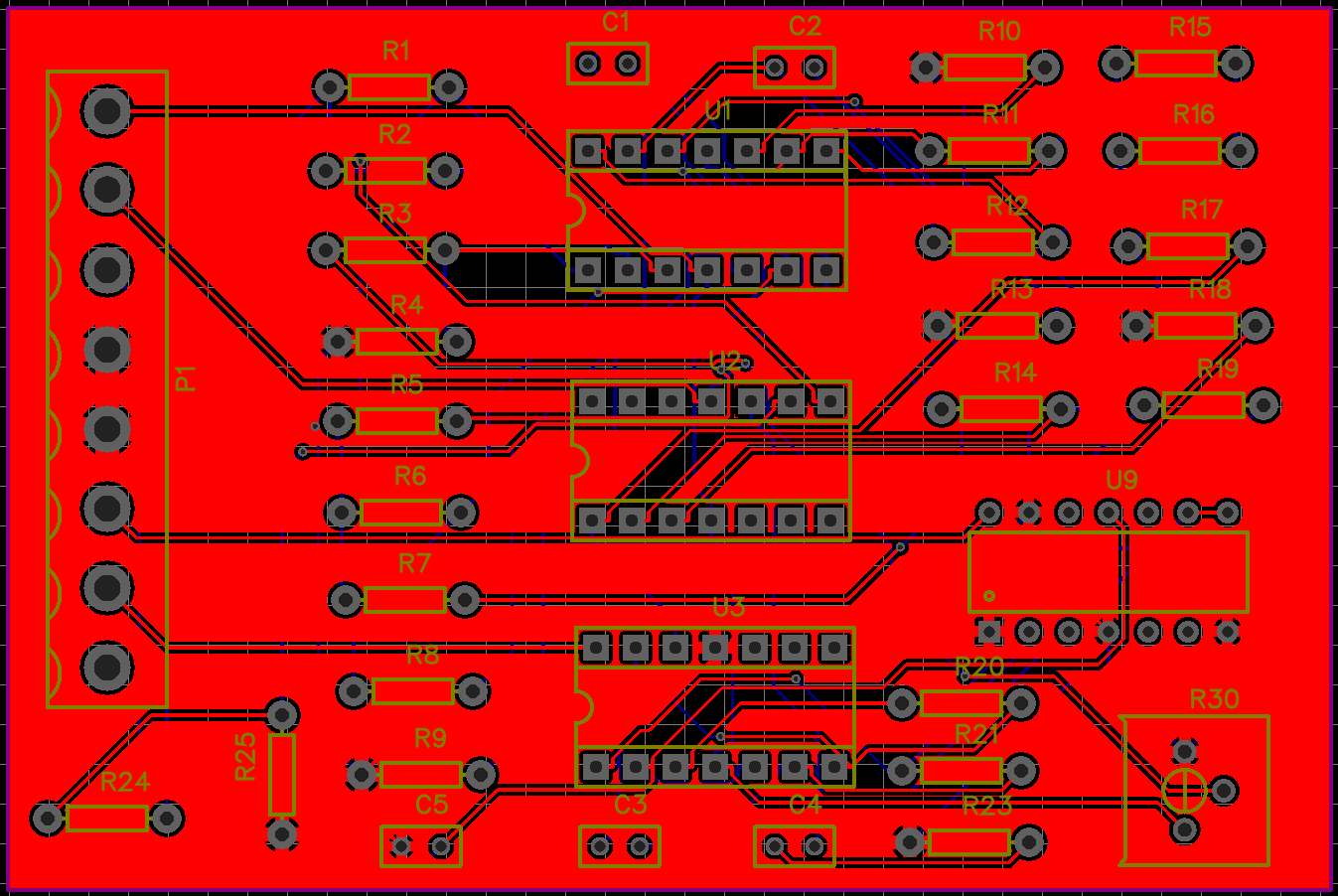
Senior Design II:

The PID schematic was changed because a triangle waveform circuit is used instead an astable multivibrator. This has the same function except that it allows for lower frequencies to be used in order to power the boiler through a solid state relay. The new schematic can be seen in figure 78. Further, an AD595 monolithic thermocouple amplifier was used the small voltage at the k-type thermocouple. This amplifies the thermocouple voltage so that the output voltage is ten times the degrees in Celsius.The thermocouple used allowed for easier testing since it had leads compatible with breadboards. It also featured a fiberglass tip for higher durability. This voltage was further amplified by a non-inverting amplifier.



*Figure 78 – Final PID Schematic Design*

The final PID PCB is shown in Figure 79. It features an 8-pin terminal block that allows interfacing with the various inputs and outputs. These are 12 V, -12 V, 5 V, ground, the positive and negative leads of the thermocouple, and the output. Three quad op-amps are placed in the center of the PCB and their accompanying components were placed nearby. One potentiometer adjusts the setpoint. The operational amplifier LM2902 was used in place of the NE5332 because of its similar characteristics as a general purpose amplifier. However, the new op-amp is a quad op-amp DIP package that allowed for easier and an operating temperature of 125°C. In addition, it was readily available.



*Figure 79 – Final PID PCB Design*

The Microcontroller PCB design was drastically altered during senior design two. We went from just implemented a simple PCB with pin hookups to a full blown setup where all the drivers and peripheral connectors are built straight into the PCB. The final sketch and design can be seen below in figure 80.

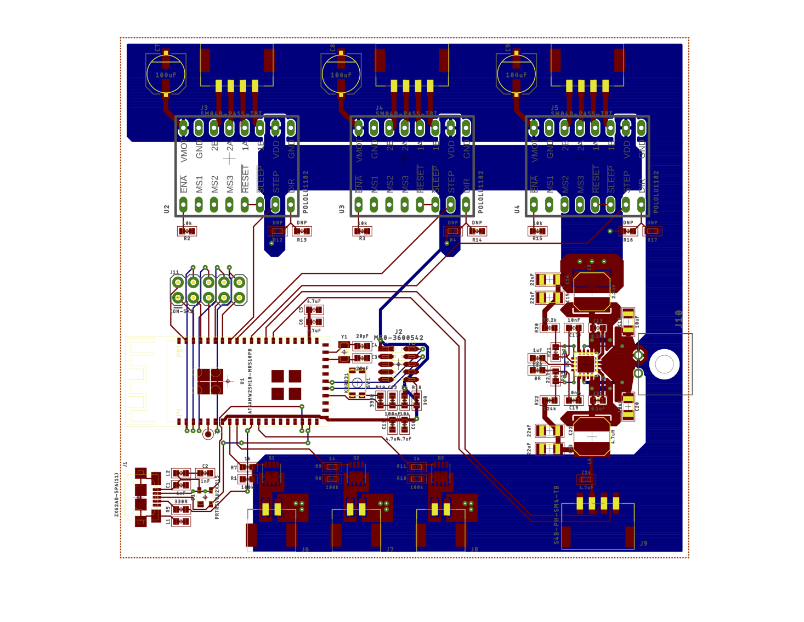


Figure 80 – *Final Microcontroller PCB Design*

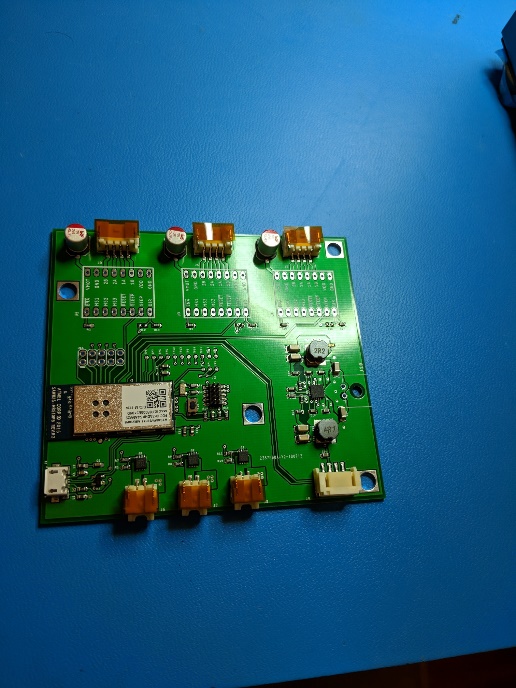


Figure 81 – *Final Built PCB Design*

**6.1.3 PCB Vendors**

The vendor used to provide the PCB is also a very important factor that comes into play when ordering a PCB. The following is a list of vendors and its details.

1. Advanced Circuits: Provide next-day shipping for $35. This is the price for students for a 60 square inch board with solder mask and top silkscreen. Since the project involves multiple printed circuit boards, this option is one of the cheapest ones if we simply cut the different parts of a board. Panelizing any board adds $50.
2. PCB Express: The end-result of the board and the process is very similar to Advanced Circuits’ process. They also have next-day PCBs without mask and quick turn around for PCBs with a mask. They are priced competitively, offer many options, and have a very detailed production service.
3. PCB Cart: They produce, assemble and test the PCB. They are based in China but have no prototyping option when ordering a single PCB. However, they claim higher production tolerances, IPC2 compliance, and have free DFM checking for up to 32 layers.

**6.2 Prototype Expectations**

The follow section will outline some of the issues and constraints that the development team could run into during the prototyping step of the system’s development.

**6.2.1 Potential Hardware Issues**

When working a project of this magnitude, there are a variety of potential hardware issues that could jeopardize the entire project. Therefore, pinpointing ahead of time some possible issues that could negatively impact our design will be vital to the success of our coffee machine.

In a project with as many parts and components such as ours, possible delays in the shipment of parts must be taken into consideration. Parts shipped from international countries are especially at risk of sudden delays. If a necessary part for a subsystem gets delayed, it can cause a bottleneck in the entire production of the project. Therefore, when choosing our components, we made sure that in the event of a part getting delayed, there will be a suitable off-brand replacement. We will also be purchasing the majority of our parts and components as soon as possible, in order to avoid wasting as little time as possible. The time to complete this project over the summer will be very limited, and we will not be able to afford wasting time being held up waiting for parts to come in.

One of the most important aspects of our project are the PCBs which will control the subsystems of the entire coffee machine. Therefore, it is extremely important to make sure our PCB design, schematics, and board layout are correct before sending them off to be manufactured. Not only in PCB manufacturing costly, it also takes a significant amount of time to be produced. If the development of our project gets halted due to a PCB defect, it will take time to pinpoint the defect, correct it, and manufacture another PCB. In a project as time limited as this, we cannot afford to have this happen. To ensure this will not happen our PCB designs have been thoroughly tested to ensure they are correct from the beginning.

Adding to the last point, even if our PCB design is perfectly designed, there is always the possibility of accidentally mishandling the board and damaging some of its functionality. When soldering parts onto the board, such as the microprocessor or the wi-fi module, it is easy to make a mistake. The possibility of making an unwanted connection, or shorting pins during the soldering process is something that should be taken into consideration.

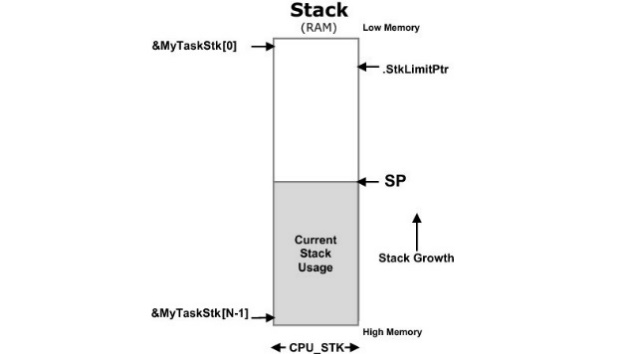
Another potential hardware issue could stem from the fact that we do not have much room to work with. Our coffee machine will only be as big as it needs to comfortably house all our internal components are electronics, to make it as efficient as practical as possible. This raise concerns over how we will distribute the space inside the machine. Parts such as the board, for example, should come nowhere near into contact with our PCBs as the high temperature of the boiler will most certainly damage the board. Deciding how to allocate all our components inside our machine will certainly be one of the most important decisions we make in our final design, and should not be taken lightly.

**6.2.2 Potential Software Issues**

When programming on an embedded system, there are a few limitations that must be considered. These limitations associated with the use of the system’s Arduino based microcontroller PCB could lead to some specific issues when it comes to how the software is written for the system. The first main issue would be that the Arduino IDE that will be used to develop some of code does not have a built-in debugger. This will make the process of debugging any potential problems even more arduous than it already is. Unfortunately, the use of serial print statements is our best bet for a somewhat efficient debugging process without the presence of a built-in debugger to show the step by step process of how of the variables are affected by the sketch being run. Conditional debugging blocks will be used to try and optimize the software debugging process as much as possible.

Another software issue that could potentially arise is integer overflow. Integer overflow is some operation in the software attempts to create a result that is outside the bit range of the processor. The system’s will be using a Cortex-M0+ 32bit ARM processor which will give us a integer limit of -231-1 to 231-1 for any signed integer. An integer calculated outside this range would result in the value being looped back to the other end of the range instead of returning the correct value. Within the scope of the current design for our system, integer overflow may not be a design constraint that will severely impact the development of the system’s software. Although, it is still something that could be a potential issue in the long run if the bit ranges of our hardware is ignored.

Stack overflow is another potential software issue that could occur during the system’s prototyping stage of development. Stack overflow occurs when the program uses up more memory than is available for the program stack. As show in Figure 82 below, the systems program stack is a Last in First Out (LIFO) buffer that keeps track of the execution of the instructions in the system’s code. With only 32KB of SRAM memory this could be a potential issue for the initial design of the system’s software. Since the system’s microcontroller will be handling quite a few subsystems, careful consideration must be taken to keep the software prototype efficient and simple in execution to prevent this buffer overflow.



*Figure 82 Program Stack Illustration*

**6.2.3 Prototype Constraints**

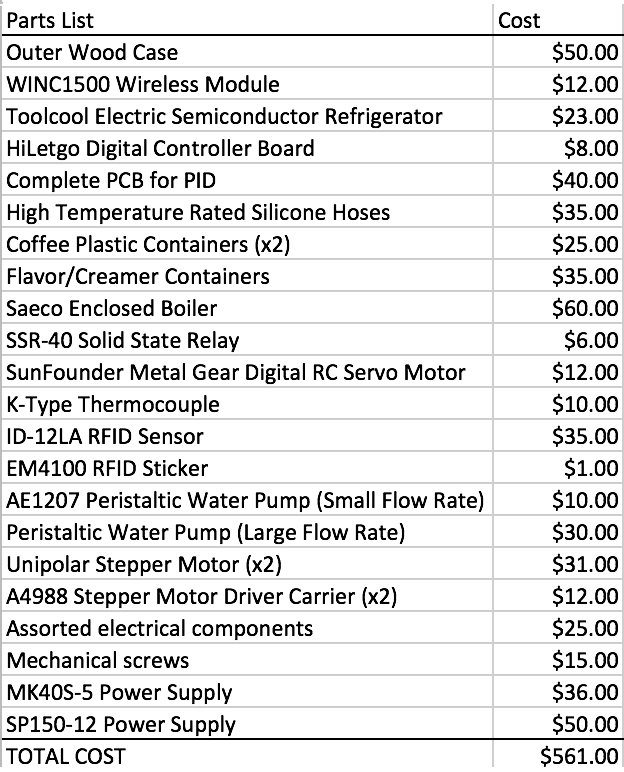
There multiple economic, ethical, health and safety constraints that must be met with the development of our prototype. In order to develop our product in the way that our original design intended will be a fairly expensive process. With our premilitary design of the system’s prototype the cost is estimated to be around $650. With the introduction of any stretch goals we intended to implement in our system, this cost will only grow. Taking in the size and construction of the systems two PCBs will be another issue for us with our current monetary restrictions. This could potentially alter the functionality of our original design and potentially prevent the ability to hand solder the components onto our PCBs. Accidental errors that could occur with the hand soldering of the boards parts could potentially lead to a damaged or unreliable PCB. To prevent this, we plan to have the components professionally soldered on to our PCB. However, our PCB design will have to be technically sound before it will be sent to a company to apply the components. Monetary restrictions may potentially not allow us to repeat this process multiple times if errors in the PCB design are discovered during the prototyping phase of development. Monetary restrictions will also be affecting the delivery time of the system. Currently peristaltic pumps will be used to send liquids throughout the system to brew the coffee. Our preliminary design includes pumps that could take up to forty-five seconds to get the water to the boiler for the largest size of coffee. Depending on the efficiency of the rest of the system, the timing issue could be mitigated. A potential solution would be on the software side of the design. The potential implementation of multi-threading would allow the designs subsystems to operate asynchronously, which if implemented correctly could greatly decrease the system’s overall delivery time.

From an ethical viewpoint, our design will have to factor in quality security for user information. User data and payment information will be stored on our servers, so property security will need to be put in place to prevent the lead of any user information that will sent over Wi-Fi communication. Health and safety concerns for our users is another constraint that will have to be accounted for in the development of the system’s prototypes. The system deals with heated surfaces and substances, so proper consideration will have to be taken to make sure all temperature sensors and wiring are accurate connected and functional.

**6.3 Parts Acquisition and Bill of Materials**

The table below shows the estimated build costs for the coffee machine. This information will be modified as progress in the design is made. The total cost nor the individual cost of any particular part is final. These decisions will be made by studying different trade-offs and how different choices compare in quality, price, ease of use, and other desirable parameters. The cost of parts that depend on unknown factors such as the PCB and other miscellaneous parts are especially subject to change. However, total costs will not exceed the maximum expenditure limit of $1000. In fact, effort will be made to keep total cost at a minimum without a major effect in quality and other desired parameters. All materials will be purchased through several online retailers, mostly Amazon and EBay. The operational amplifiers will be bought directly from Texas instruments and the electrical components will be purchased from Digi-key. The boiler itself will be bought from a parts distributor known as Saeco. The boiler chosen is also used as a replacement part in many other commercial coffee machines.

*Table 30 – Parts List*



**7 Testing**

Different equipment will be used to for testing in general. A Tektronix DPO 4034B Digital Phosphor Oscilloscope will be used to verify that electronics are producing the expected output. A Tektronix DMM 4050 Multimeter will be used to test currents and voltages. The Keithley 2230-3-1 Triple Channel DC Power Supply is the power supply of choice for this project. Testing will mainly take place in the Senior Design Lab. This lab contains all the testing equipment needed to carry out this project. Although every part that should be needed is bought through components suppliers such as Digi-Key, assorted components supplied by the lab the might be used. In addition, all electronics prototypes will be tested using a set of three 830 Point Solderless Breadboard from Elegoo and 22 gauge wires for all connections. Parts that do not require the use of electronics will be tested and built using the Texas Instruments Innovations Lab. For example, the exterior of the machine will involve basic woodwork while some connections inside the machine require use of mechanical parts that deal with the transportation of water. This lab also supplies a smaller collection of assorted electronic components.

**7.1 Hardware Testing**

The following sections will go into detail on how the physical hardware of the system will be test to ensure the system will be ready to deliver to the customers.

**7.1.1 Microcontroller Testing**

Test Case Environment

The first round of testing will be conducted using an Arduino MKR1000 development board. Each test case will involve testing individual components of the board to ensure everything is working properly. The second round of testing will begin, once it has been confirmed that the board has no major faults. The second round of testing will be done with our microcontroller embedded PCB. The second round of microcontroller testing is more integral to testing our design since it will allow us to find any faults in our PCB if there are any.

Description of Individual Test Cases

Test Case Reference Number: TC1

* Test Objective: Power board through Vin
* Test Description: The test will start with using a power supply to apply 5V to the boards VIN pin. If the board shows signs of powering on then the test case passes. If the test case fails immediate attention will need to be put toward the microcontroller because a fault in powering our MCU is highly detrimental to the system’s success.
* Test Conditions: None
* Expected Results: The board will power up.

Test Case Reference Number: TC2

* Test Objective: Power board through USB
* Test Description: The test will start by plugging the power cable into the board’s USB port. The test case will pass if the board shows signs of powering up. If the test case fails immediate attention will need to be put towards troubleshooting why the USB connection to the board is broken.
* Test Conditions: None
* Expected Results: The board will power up.

Test Case Reference Number: TC3

* Test Objective: Check clock accuracy
* Test Description: Start the controllers Real Time Clock(RTC) for a specified amount of time. If the time duration matches that of your desired time the test case has passed.
* Conditions: Basic setup of board is complete
* Expected Results: The RTC is accurate

Test Case Reference Number: TC4

* Test Objective: Check input functionality of Digital I/O pins
* Test Description: Using the Arduino IDE, set one of the boards digital I/O pins to act as a digital input. Then connect this pin to the system’s boiler sensor to see if the data is taken in correctly.
* Test Conditions: The boiler subsystem should be setup and connected to the PCB/Board.
* Expected Results: The Digital I/O pin reads in the temperature sensor data.

Test Case Reference Number: TC5

* Test Objective: Check output functionality of Digital I/O pins
* Test Description: Using the Arduino IDE, set one of the boards digital I/O pins to act as a digital input. Then connect this pin to a simple LED and have pin send a signal to light up the LED.
* Test Conditions: Basic setup of the board should be complete.
* Expected Results: The LED is lit.

Test Case Reference Number: TC6

* Test Objective: Check if the PWM pins duty cycle is accurate
* Test Description: Use the built in library’s function analogWrite() to set the duty cycle of a pin with PWN functionality. Use an oscilloscope to check that the pin is producing the correct waveform for the selected duty cycle. Repeat this test multiple times with varying waveform frequencies.
* Test Conditions: Basic setup for the board should be complete.
* Expected Results: The correct waveforms are displayed on the oscilloscope.

**7.1.2 RFID Scanner Testing**

Test Case Environment

The first round of testing will be conducted using an Reader ID-12LA (125 kHz) connected to the Arduino MKR1000 development board. The RFID tags that the system uses will also need to be tested in each step to verify they are functioning correctly. Each test case will resolve around testing whether the RFID reader is properly hooked up to the microcontroller, as well as making sure it can properly read a RFID tag. Once it has been confirmed that the RFID reader is operating as intended and all test cases have been passed, the second round of testing will begin. The second round of testing will be done with our RFID scanner connected to the microcontrollers embedded in the system’s PCB to test the integrity of the of the PCB’s design.

Description of Individual Test Cases

Test Case Reference Number: TC1

* Test Objective: Make sure RFID module is wired correctly.
* Test Description: THE ID-12LA RFID reader will be wired to the microcontroller as the modules documentation states. Then a digital multimeter will be used to measure the current across the pin that the RFID reader is connected to. If there is a current being sent through the pin then the test case passes.
* Test Conditions: Microcontroller testing has been completed.
* Expected Results: The multimeter will read a non-trivial value when the current is measured from the microcontroller’s pin.

Test Case Reference Number: TC2

* Test Objective: Verify the RFID tags are read by the reader.
* Test Description: THE ID-12LA RFID scanner will be properly hooked up to our MKR1000 development board or PCB. A sketch will be created to test if the pin that is connected to the RFID reader receives any input value. If a value is received the tag has proven to be operable and the test case passes.
* Test Conditions: RFID scanner properly connected to board.
* Expected Results: The tag will be read correctly and the LED will blink

**7.1.3 PID Controller Testing**

This section will cover how to test the PID controller once it is built in a breadboard to make sure the design follows the expected outputs. Firstly, all resistances and measurable components should be measured to ensure they are within a tolerable range. Each stage of the PID controller will be tested using an oscilloscope. Although testing of each amplifier is not necessarily required, one can confirm each one works if it behaves as expected. Each individual output should match the output provided in the simulations presented. The output response as each gain term changes will have to be studied carefully. Potentiometer trimmers will be set to the provided schematic’s resistance before starting to change any of the gain terms. These circuits will all bested individually and then everything will be combined. If an unexpected output occurs but the P, I, and D terms seem correct, different combinations of terms will used to find the issue. For example, a PI controller will be used to simplify troubleshooting before moving on the PID controller. The error amplifier and the astable multivibrator are two things that can be tested easily using an oscilloscope. Before moving on to the full PID controller, the error amplifier will need to be tested using inputs from a function generator. Then it will be tested with inputs from the thermocouple sensor and an LED. The PWM output will also be tested using an oscilloscope and multimeter to measure the low and high voltages. The relationship between duty cycle of the PWM produced by the controller and the PID parameters will be studied closely. The thermocouple will be tested at lower temperatures at first. Temperature will be increased slowly until the desired temperature required is reached to ensure that it works adequately. Another temperature sensor will be used to measure its accuracy as temperature increases and decreases.

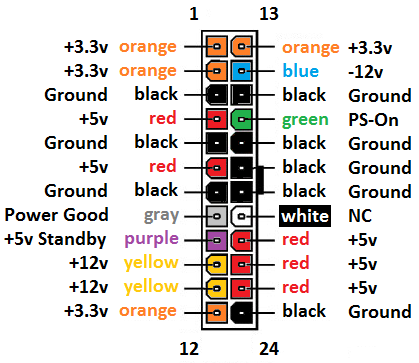
The solid-state relay is going to be tested by using a function generator before moving on to high power application to ensure safety. This also includes testing how a pulse width modulated signal affects the input. The test will be repeated for the pulse width modulated signal obtained from the output of the PID controller. Testing with mains power will only occur after all of these tests are successful. Many additional safety precautions will need to be kept in mind before moving on including insulated gloves. Before turning on anything, the outputs will be measured using an oscilloscope and all connections will be checked to make sure everything is wired adequately. If anything starts burning or smoke starts coming out of a component, power will be turned off immediately.

For testing a PCB, one should first inspect it visually for any obvious defects. After this step, there are two tests that can be done ensure the PCB is working as designed. The capacitance test will test for opens and shorts by sending a charge on the trace. One then probes each trace and measures the induced capacity. The second test is the resistance test. It is a way of measuring the resistance in ohms as the current flows through a trace. This testing procedure also applies for other PCBs

**7.1.4 Power Supply Testing**

Testing to see if our power supplies are working properly is one of the most important testing we must perform in our design. If our power supplies are defective, none of the electronics inside our coffee machine will be able to function properly, and it could even pose a fire hazard if the power supply is defective enough.

There are a couple of ways to determine if the power supply is behaving as expected. The first and crudest one is called the paperclip test. To perform this test, we must disconnect the power supply from its input power and power it off. On the 24-pin connector, we locate the green and black wires going into pins 16 and 17 respectively, and attach each end of the paperclip to them. After this we reconnect the power supply back to mains power and turn it on. If the power supply fan turns on, that means it is working properly. This works because the paperclip “tricks” the power supply into thinking there is a load connected to it. The figure below shows the standard 24-pin ATX that the majority of power supplies use.



*Figure 83 - 24-pin ATX connector*

The second, and more proper way to test a power supply is through a multimeter. It is effectively an extension of the paper clip test. We disconnect the power supply from any input power and turn the power supply off. Then create a short with pins 15 and 16 with a small piece of wire, and turn the power supply back on. Now we will use the multimeter to test the voltages across the pins of the 24-pin ATX connector. The 24-pin connector has different voltage lines, and to be sure they are working properly we will test each of them. The negative probe of the multimeter will go to any of the ground pins, and the positive probe will be used to test the voltages across the 3.3V, 5V, and 12V lines. If the multimeter reads the correct voltages, then the power supply is working as expected and is ready to be used in our project.

**7.1.5 Boiler Testing**

One needs to take every precaution when testing the boiler that will be used for this project as the temperature can get very hot. The fact that it will not be enclosed in a machine while testing makes these temperatures it even more dangerous. Another reason why this is dangerous is because the boiler is being powered by AC mains voltage. The first thing to test the boiler before turning it on is to check its resistance. Although it is can be printed on the boiler, it is better to measure it. The expected resistance can be found by squaring the voltage used and dividing by power. A multimeter can be used to measure the resistance of the boiler’s leads. A higher resistance reading than the calculated value suggests that the boiler will not heat to its maximum capabilities. If resistance is lower, it means that there is too much or that the part of element is being shorted. Check for continuity using the multimeter before proceeding. When testing the boiler, make sure it is unplugged and that it does not have any water inside. The first thing to test the boiler with is to connect it to a wall outlet directly. If this is successful, the next step is to test the different how a solid-state relay and pulse width modulated signal can be used to control it. The astable multivibrator circuit used in the final PID schematic design can be used to generate a PWM signal. Alternately, a 555 timer in astable mode can also be used. Different duty cycles will be examined along with a temperature sensor to observe the overall relationship between duty cycle and temperature. This cannot be done accurately because of the difficulty in measuring temperature values.

**7.2 Software Testing**

The purpose of the following testing sections is to lay out a plan on how to test the functionality and integrity of all the code used for our system. This includes all the code to control our microcontroller as well as the code written to develop the systems mobile app. Most of the following tests will be completed in multiple phases. The first phase of testing will be performed on the Arduino MKR1000 development board. The second round of testing will be with the system’s PCB to test the integrity of its design and prove it provides the same functionality as the development board did. The reason the test was broken down into these phases is so that we can tackle any problems with the software early on and without concern that the error could be due to a fault in our PCB or subsystem design. Knowledge of the error being software base can help the debugging process become as clear and concise as possible. The stopping point for the systems software testing process will be once all our test cases have been passed. All of test cases passing will signify that all the main objectives of our design have been met and our system will be considered in a deliverable state for its future users.

**7.2.1 Microcontroller Software Testing**

Test Case Environment

The first round of testing will be conducted using an Arduino MKR1000. Each test case will involve running individual bits of the microcontroller’s code to confirm that the microcontroller is correctly sending and receiving data from the rest of our system. Once it has been confirmed that the code has all the subsystems interacting correctly and all test cases have been passed, the second round of testing will begin. The second round of testing will be done with our system connected to the microcontrollers embedded in the system’s PCB to test the integrity of the of the PCB’s design.

Description of Individual Test Cases

Test Case Reference Number: TC1

* Test Objective: Test Wi-Fi communication of the MKR1000
* Test Description: Ensure that the Arduino Wi-Fi 101 library is included. Then run some simple code that scans for the available Wi-Fi networks and attempt to connect to one. If connection is successful the test case will pass.
* Test Conditions: The MCU must have been through basic setup and have all the correct libraries included.
* Expected Results: The board will be able to see the Wi-Fi connects and be able to connect to one.

Test Case Reference Number: TC2

* Test Objective: See the MKR1000 receives data from the RFID receiver.
* Test Description: Start by scanning a RFID tag that has a known ID. If data is received from the RFID receiver, the microcontroller will then compare the received value to the known ID
* Test Conditions: A properly functioning RFID receiver connected to the MKR1000.
* Expected Results: The correct value should be received.

Test Case Reference Number: TC3

* Test Objective: The MKR1000 properly operates the stepper motors.
* Test Description: Have the MKR1000 send multiple signals to move the motors by varying amounts of steps. After each signal, check to verify that the correct amount of grounds is dispensed.
* Test Conditions: The dispenser subsystem must be setup and connected to the microcontroller.
* Expected Results: The correct amount of grounds will be dispensed.

Test Case Reference Number: TC4

* Test Objective: See if the MKR1000 properly controls the boiler.
* Test Description: Have the MKR1000 signal to the boiler to take in water. If the boiler correctly heats and dispenses the hot water, the test passes.
* Test Conditions: The boiler subsystem should be setup and connected to the PCB.
* Expected Results: The boiler heats and dispenses water when it is signaled.

Test Case Reference Number: TC5

* Test Objective: See if the MKR1000 properly receives data from the app.
* Test Description: Have the app send a drink order to the MKR1000 over Wi-Fi and have the microcontroller read and do a check to see whether the correct user’s info was received.
* Test Conditions: The app must be completed and have route information to contact the MKR1000.
* Expected Results: The correct info should be received and displayed.

Test Case Reference Number: TC6

* Test Objective: See if the MKR1000 properly interacts with the database.
* Test Description: Use the MKR1000 to call get requests on our database to fetch the user’s stored data. If the MCU receives the correct user’s drink preferences and a hash of the payment info, then the test passes.
* Test Conditions: The database must be setup and have user’s information stored.
* Expected Results: The correct user’s info should be obtained and displayed.

Test Case Reference Number: TC7

* Test Objective: See if the MKR1000 properly controls the pump to dispense milk.
* Test Description: Have a user order a drink that contains milk. The microcontroller will then send out a signal to the pumps controlling the milk dispensary. Proper connection will be confirmed if milk is dispensed are the user’s coffee has been brewed.
* Test Conditions: The Milk dispensary subsystem must be setup and connected to the microcontroller.
* Expected Results: The pump should add milk to the user’s coffee.

Test Case Reference Number: TC8

* Test Objective: See if the MKR1000 properly handles payment information.
* Test Description: A get request will be called on the database asking for a specific user’s payment info. If anything but a hash of the data is returned, the test fails.
* Test Conditions: The database must be setup and have users stored in the system.
* Expected Results: A hash of the user’s payment should be received.

**7.2.2 Database/Server Testing**

Description of Test Environment

The purpose of these tests is to test the integrity of our database and the servers they will be on. The first wave of testing will be run on various individual systems which include our laptops and desktops. These tests will be performed on various software environments which include different browsers such as Chrome and Safari, and programs that facilitate communications with our database such as Postman and Robo 3T. Once the user app is completed the second wave of these tests will begin to verify connection to the database and the apps functionality.

Description of Individual Test Cases

Test Case Reference Number: TC1

* Test Objective: See if a user can login to the system
* Test Description: Start with an empty users model. First select a random username and password, and then try and login with them. If login succeeds, the test fails. Now enter a user into the database and try and login with the new credentials. If login succeeds, the test passes, if not the test fails.
* Test Conditions: The database and users model must be setup
* Expected Results: The login will first be denied, then accepted.

Test Case Reference Number: TC2

* Test Objective: See if system allows the creation of new users.
* Test Description: Make a call to the register() function and attempt to add a new user to the database with all the required credentials. The next step will be use Robo 3T to verify that the new user has been stored in our database
* Test Conditions: The database and users model must be setup.
* Expected Results: A new user should be created.

Test Case Reference Number: TC3

* Test Objective: See if the system properly secures passwords.
* Test Description: Call a post request to add a new user to the database. Check the return value of this command to confirm that the password has been hashed and not sent back in the return value of the communication.  Then call a get request to make sure the password is not passed as a result.
* Conditions: The database and users model must be setup.
* Expected Results: The user’s password should be hashed, and the password should not be returned when a get request is made.

Test Case Reference Number: TC4

* Test Objective: See if database system properly fetches information from our database models.
* Test Description: Call a get request on all the models to get all the allowed information for each model.
* Test Conditions: The database should have at least one entry in it.
* Expected Results: The requested model information will be returned to the user.

Test Case Reference Number: TC5

* Test Objective: See if database system properly stores entered model info.
* Test Description: Call post request with new patient/user/etc info. Then call a get request on their IDs to see if the inserted info matches the retrieved info.
* Test Conditions: The database and user models must be setup.
* Expected Results: The inserted and retrieved information will match.

Test Case Reference Number: TC6

* Test Objective: See if database system properly deletes a model from the database.
* Test Description: Call a delete request on a current model’s ID to remove the object from the database. Then call a get request with the same ID to verify the object has been removed from the database.
* Test Conditions: The database must have at least a single model stored in the database.
* Expected Results: The specific model will be removed from the database.

Test Case Reference Number: TC7

* Test Objective: See if database system properly edits a model from the database.
* Test Description: Call a patch request on a current model’s ID and send the model the information to be changed. The return value of the patch request should contain the model with its updated info. A get request should then be called to verify only information that is allowed to be changed has been edited in the database.
* Test Conditions: The database must have at least a single model store in the database
* Expected Results: The database should have the correct information edited on each of the models.

Test Case Reference Number: TC8

* Test Objective: Test app interaction with database/server
* Test Description: Navigate through the app’s user interface, clicking every button to see if it behaves as expected. If the app crashes or a button does not perform the correct task, the test fails.
* Test Conditions: All other Test Cases in this Module must have passed testing.
* Expected Results: The app should interact with our server as expected with no crashes.

Test Case Reference Number: TC9

* Test Objective: Test the QR code registration
* Test Description: Navigate through the app’s user interface, and find the option that allows the user to add a new RFID tag to the account. Select this option and go through the procedure to finalize the addition of the tag to your account. Then check the database to see that the new tag has been registered to the correct user. The test case passes if the new tag is stored in the user’s model.
* Test Conditions: All other Test Cases in this Module must have passed testing.
* Expected Results: The new tag ID should be stored in the database under the user’s model.

**7.2.3 RFID Testing**

Test Case Environment

The first round of testing will be conducted using an Reader ID-12LA (125 kHz) connected to a desktop computer to confirm that the module interacts correctly with the systems. Each test case will involve running our testing our RFID scanners code to see if it is correct correctly sending and receiving the data is receives from the tags. Once it has been confirmed that the code is performing as intended and all test cases have been passed, the second round of testing will begin. The second round of our code’s tests will be done on the MKR1000 breakout board to confirm our software runs correctly on the microcontroller. The third round of testing will be done with our RFID scanner connected to the microcontrollers embedded in the system’s PCB to test the integrity of the of the PCB’s design.

Description of Individual Test Cases

Test Case Reference Number: TC1

* Test Objective: RFID code test on computer.
* Test Description: THE ID-12LA RFID scanner will be hooked up to a computer and have all the proper firmware installed. The Arduino IDE will then be opened and the Port and Serial Monitor tools will be utilized to view the info that our code receives. A tag with a known identification value will be scanned and its value should be displayed in the serial terminal. If not, the test fails.
* Test Conditions: RFID scanner firmware has been installed.
* Expected Results: The correct tag value will be displayed in the Arduino IDE’s serial terminal.

Test Case Reference Number: TC2

* Test Objective: RFID code test on MKR1000/PCB.
* Test Description: THE ID-12LA RFID scanner will be properly hooked up to our MKR1000 development board or PCB. A tag with a known identification value will be scanned and if the value is recognized by the code, the boards LED will blink.
* Test Conditions: RFID scanner properly connected to board.
* Expected Results: The tag will be read correctly and the LED will blink

Test Case Reference Number: TC1

* Test Objective: Check RFID tag recognized when passed to database.
* Test Description: THE ID-12LA RFID scanner will be properly hooked up to our MKR1000 development board or PCB. A tag with a known identification value will be scanned and its value will try to be identified in the database. If the tag is recognized a LED attached to our board will blink to confirm the success.
* Test Conditions: Database is setup, and RFID scanner properly connected.
* Expected Results: The tag will be read correctly and the LED will blink

**8 Administrative Content**

**8.1 Division of Labor**

Below in Table 8.1-1 is general guideline for how the tasks were divided amongst the group members, with most tasks having a primary and secondary team member responsible for its completion.

*Table 31 - Division of Labor*

|  |  |  |
| --- | --- | --- |
|  | Primary | Secondary |
| RFID Module | Alessandro |  |
| Hardware Programming | Dylan | Alessandro |
| PID Controller | Daniel |  |
| Coffee Dispenser | Dylan |  |
| Boiler | Daniel | Guilherme |
| Power Supply | Guilherme |  |
| Casing | Alessandro |  |
| Refrigerator/Pump | Guilherme |  |
| Database/Server | Alessandro | Dylan |
| Mobile Application | Alessandro | Dylan |
| PCB Design | Daniel/Guilherme |  |

**8.2 Project Milestones**

The following is a list of action items and the date each should be completed by. This will serve as a way to keep track of progress and ensure that all aspects of the project and the report are complete by the due date. Failure to comply with any of these due dates will delay the process and result in a limited amount of time towards the end of the semester.

February 4th : Get an idea of what similar projects have done

February 6th : Decide on all features that will be implemented

February 20: In-depth research of similar projects

February 28: Decide what technologies we will use to achieve objective

March 20: Table of Contents Completed

March 29: Model electronics on breadboard

April 9: First 60 pages of report due

April 22: Have draft of final report due ready

April 27: Turn in Final Report

May 2: Buy all parts needed

May 29: Test everything and become familiar with sensor functionality

June 14: Have all parts successfully communicating with each other

June 25: Order preliminary PCB design

July: Have app interface finished

July 15: Have system working together

July 20: Finish Testing and optimize timing issues if needed

August 1: Fix any last minute issues and prepare presentation

**8.3 Budget and Finance**

The budget for this project was established to be no more than $1000. However, the goal of this project was to design and build a working unit of the coffee maker without the need to purchase premium parts. In other words, each part chosen was selected with the purpose of having the coffee maker unit work efficiently without sacrificing on costs. Materials will be ordered as needed during the building process. The team member in charge of research and design of a specific part is responsible for purchasing, testing, and implementing their part. They are also responsible of keeping track of everything they buy and keep receipts. They are also responsible for requesting refunds if need be. More expensive parts such as the boiler require approval from all team members before purchase. This is because each team member has different preferences and specialized research on how the boiler chosen could impact their part of the project. Both of these reduce the chances of returns that could potentially delay the building process. After all materials are purchased, including used and unused materials, a table will be made with the product, price, and buyer. The total cost will then distribute among the four team members. Since the estimated cost is $685, each team member will be paying $171.25 in total. All expenses will be divided using the app Splitwise. This app allows for every expense to be divided equally among the team as long as the team member who purchased the item records the purchase. Evidence of purchase will need to be sent to other team members for verification. At the end of the project, reusable parts such as the boiler will be sold through Ebay. All profit will be split among all four team members. If a team member decides to keep the coffee machine, he would need to pay the profit the other team members would have gotten in order to keep the machine. In the case that a person not affiliated with the group is interested in buying our specific coffee machine, they will be charged twice the cost of the machine to account for all the work and planning that came into producing this machine. The buyer would be free to collect earnings from the coffee machine. However, they would be solely responsible for maintaining the machine. They should also understand that the cost of the machine does not include servicing. No additional costs for servicing will be sold under no circumstances.

The unit was not designed with the end-goal of mass production in mind. The main goal was to show a proof of concept of implementing a convenient coffee vending machine service. A coffee machine was still built for demonstration of this service. If the coffee machine were to be mass-produced, different modifications will need to be made. For example, some parts such as the water storage would need to be implemented with a water reservoir. Another thing that will need to be changed is the analog PID controller. This controller was built purely for demonstration purposes. A digital PID controller would allow for more fine-tuning of the PID parameters and be a lot more reliable than the current design. Other things that would need to be changed are the exterior design and the machine’s ease of use. A more marketable product that could result from this senior design is a system that can make the RFID payment system implemented in the project into any coffee machine. More research would need to be done in order to determine market potential since this idea would infringe on the warrantee of other coffee machines and could have other repercussions.

**9 Conclusion**

This report goes over the research, design, and testing stages required for the development of the Moka coffee machine. Extensive research was conducted to ensure that all aspects of the building process were accounted for. This will ensure a smooth transition from the development stage to the building process. All of the design details that went into this project have the objective of helping produce a successful outcome while abiding by the constraints. This paper was written entirely by its team members. Credit was given through references and in-text citations to other material that was proven useful.

If successful, the Moka coffee machine will be a great product ideal for small companies and coffee shops. Simply by using an app, users will be able to select their preferred coffee and pay instantly without having to take out their wallets. Alternatives to the Moka coffee machine are premium high-quality coffee machines. However, the Moka coffee machine offers an automated system for selling coffee with a reduced cost. This product creates a new kind of product for commercial coffee machines.

**Appendix A References**

[1] Agarwal, T. (2018). Different Types of Wireless Communication Technologies. Buy Electronics & Electrical Projects in the United States. Retrieved 1 April 2018, from https://www.efxkits.us/different-types-of-wireless-communication-technologies/

[2] American Society for Testing and Materials. (n.d.). Standard Guide for Heated System Surface Conditions That Produce Contact Burn Injuries. Retrieved from <http://arnisco.com/astm/PDF/C1055.PDF>

[3] Arduino MKR1000 WIFI. (2018). Store.arduino.cc. Retrieved 1 April 2018, from <https://store.arduino.cc/usa/arduino-mkr1000>

[4] Atmel ATmega328/P Datasheet. (2016, November). Retrieved from http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P\_Datasheet.pdf

[5] Atmel SAM D21E / SAM D21G / SAM D21J. (2018). Cdn.sparkfun.com. Retrieved 1 April 2018, from

https://cdn.sparkfun.com/datasheets/Dev/Arduino/Boards/Atmel-42181-SAM-D21\_Datasheet.pdf

[6] Black Bear Coffee Micro Roastery. (n.d.). How to Brew "The Perfect Cup" of Coffee. Retrieved April 9, 2018, from https://blackbearcoffee.com/resources/87

[7] Boggan, P. (2013, October 15). Send the Monkey a Message with Multipeer Connectivity. Retrieved from https://blog.xamarin.com/send-the-monkey-a-message-with-multipeer-connectivity/

[8] Cannon, L. (n.d.). Recommended C Style and Coding Standards. Retrieved April 9, 2018, from https://www.doc.ic.ac.uk/lab/cplus/cstyle.html

[9] C Coding Standard. (n.d.). Retrieved April 9, 2018, from https://users.ece.cmu.edu/~eno/coding/CCodingStandard.html#guards

[10] Chacon, S., & Straub, B. (2014). Pro Git. Berkeley, CA: Apress

[11] DeLisle, J. (2017, March 14). What's the Difference Between IEEE 802.11af and 802.11ah? Retrieved from http://www.mwrf.com/active-components/what-s-difference-between-ieee-80211af-and-80211ah

[12] Desborough, L., & Miller, R. (2002). Chapter Ten: PID Control. Retrieved from http://www.cds.caltech.edu/~murray/books/AM08/pdf/am08-pid\_22Feb09.pdf

[13] Detecting Task Stack Overflows - µC/OS-III Documentation - Micrium Documentation. (2018). Doc.micrium.com. Retrieved 1 April 2018, from https://doc.micrium.com/display/osiiidoc/Detecting+Task+Stack+Overflows

[14] Easthope, A. (2017, May 10). Brew Temperature and its Effects on Espresso. Retrieved April 9, 2018, from https://www.fivesenses.com.au/blog/brew-temperature-and-its-effects-on-espresso/

[15] Electronic Tutorials. (2018, March 02). Solid State Relay or Solid State Switch. Retrieved from <https://www.electronics-tutorials.ws/power/solid-state-relay.html>

[16] ESP32 Datasheet. (2018). Espressif.com. Retrieved 1 April 2018, from https://www.espressif.com/sites/default/files/documentation/esp32\_datasheet\_en.pdf

[17] Fisher, T. (n.d.). Use a Multimeter to Test Your Power Supply. Retrieved from https://www.lifewire.com/how-to-manually-test-a-power-supply-with-a-multimeter-2626158

[18] Galco. (n.d.). How Relays Work. Retrieved from http://www.galco.com/comp/prod/relay.htm

[19] How To Control Stepper Motor with A4988 Driver and Arduino. (2018). HowToMechatronics. Retrieved 1 April 2018, from https://howtomechatronics.com/tutorials/arduino/how-to-control-stepper-motor-with-a4988-driver-and-arduino/

[20] How to implement PID controller! (2010, December 1). Retrieved April 10, 2018, from http://mechatronicsheaven.blogspot.com/2010/12/how-to-implement-pid-controller.html

[21] Instructables. (2017, September 22). DIY Portable Mini Refrigerator. Retrieved from http://www.instructables.com/id/DIY-Portable-Mini-Refrigerator/

[22] Kanhar, S. R. (2016, July 11). OPAMP REALIZATION AND PID CONTROLLER FABRICATION. Retrieved from http://ethesis.nitrkl.ac.in/8015/1/2016\_BT\_Soumya\_112EE0232.pdf

[23] McKinney, C. (2014, April 02). How a PID works. Retrieved April 10, 2018, from https://www.youtube.com/watch?v=qGURvZqZazo

[24] O'Dwyer, A. (2005, January 1). PID control: The early years. Retrieved from https://arrow.dit.ie/cgi/viewcontent.cgi?referer=https://www.google.com/&httpsredir=1&article=1088&context=engscheleart

[25] Peristaltic pump. (2018, April 24). Retrieved from https://en.wikipedia.org/wiki/Peristaltic\_pump

[26] Pololu - A4988 Stepper Motor Driver Carrier. (2018). Pololu.com. Retrieved 1 April 2018, from https://www.pololu.com/product/1182

[27] Printed Circuit Board (PCB) Design Issues. Retrieved from https://www.pcbway.com/blog/Engineering\_Technical/Printed\_Circuit\_Board\_PCB\_Design\_Issues.html

[28] Radio-frequency identification. (n.d.). In Wikipedia. Retrieved April 9, 2018, from https://en.wikipedia.org/wiki/Radio-frequency\_identification

[29] RFID Standards. (n.d.). Retrieved from https://www.impinj.com/about-rfid/rfid-standards/

[30] Soldered Electrical Connections. (2018). Nepp.nasa.gov. Retrieved 1 April 2018, from https://nepp.nasa.gov/docuploads/06AA01BA-FC7E-4094-AE829CE371A7B05D/NASA-STD-8739.3.pdf

[31] Texas Instruments. (n.d.). NE5532x, SA5532x Dual Low-Noise Operational Amplifiers. Retrieved from http://www.ti.com/lit/ds/symlink/ne5532.pdf

[32] Vapor-compression refrigeration. (2018, April 14). Retrieved from https://en.wikipedia.org/wiki/Vapor-compression\_refrigeration

[33] Wattco. (n.d.). ELECTRIC IMMERSION HEATERS: FREQUENTLY ASKED QUESTIONS. Retrieved April 9, 2018, from https://www.wattco.com/2015/08/electric-immersion-heaters-frequently-asked-questions/

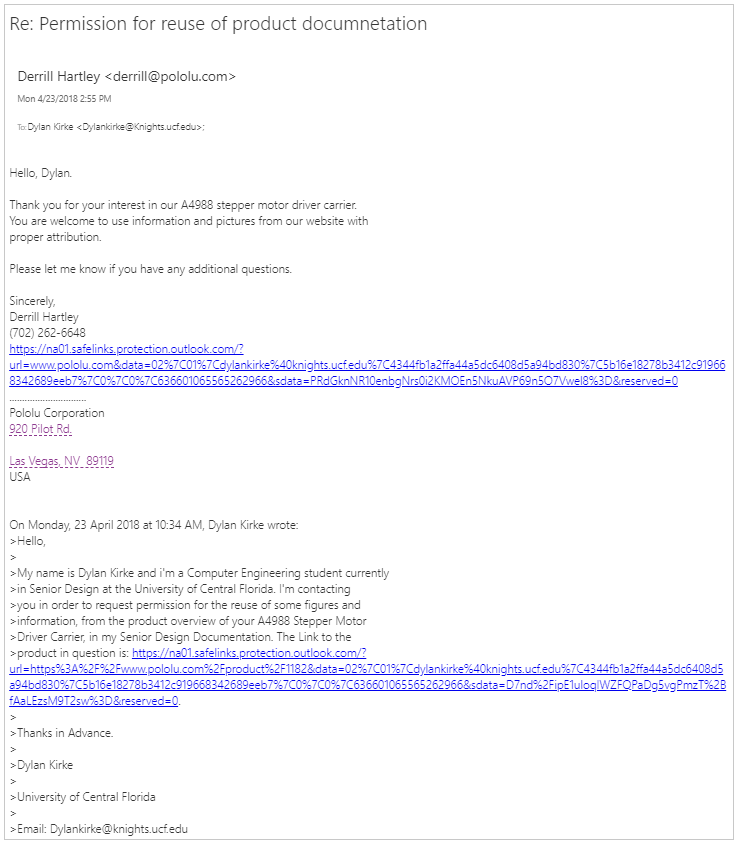
[34] What is RFID? (n.d.). Retrieved April 9, 2018, from https://www.epc-rfid.info/rfid

[35] What is Switch Mode Power Supply? (2014, September 10). Retrieved from https://www.sunpower-uk.com/glossary/what-is-switch-mode-power-supply/

[36] Woodford, C. (2017, November 11). How does Bluetooth work? Retrieved April 9, 2018, from http://www.explainthatstuff.com/howbluetoothworks.html

**Appendix B Permissions**

Polulu Robotics and Electronics



RobotShop.com

