Final Report for ECE 445, Senior Design, Fall 2018

Laundry Alarm

Project No. 8

TA: Hershel Rege

Marcin Horwat

Michelle Lamblin

Thomas Mayer

Abstract

This document outlines the design and construction process of the Laundry Alarm device and its performance validations. This device will be able to sense the current draw of a washer or dryer unit that is plugged into the device. The device will notify the user when their wash or dry cycle is complete by sending them a notification to their computer or mobile device. The device will also monitor leakage current and show that data to the user on their computer or mobile device.

Table of Contents

1. INTRODUCTION 1
1.1 Objective 1
1.2 Background 2
1.2 High-Level Requirements 3
2 DESIGN 4
2.1 Physical Design 4
2.2 PCB Design 5
2.3 Block Diagram 6
2.4 Control Unit 7
2.4.1 Microcontroller 8
2.4.2 Wi-Fi Module 9
2.5 Software 9
2.5.1 User Interface 9
3 DESIGN VERIFICATION 11
3.1 Power 11
3.1.1 AC/DC Power Supply 11
3.1.2 Voltage Regulation 11
3.1.3 Current Transformers 11
3.1.4 Leakage Current 11
3.2 Control Unit 12
3.2.1 Microcontroller 12
3.2.2 Wi-Fi Module 12
3.2.3 Android Application 12
4. COSTS 13
4.1 Parts 13
4.2 Labor 13
5. CONCLUSION 14
5.1 Accomplishments 14
5.2 Uncertainties 14
5.3 Ethical considerations 15
5.4 Future work 15
REFERENCES 16
APPENDIX A 17
Requirement and Verification Table 17
APPENDIX B 19
Cost Analysis Table 19
APPENDIX C: 20
Photo References for Software and User Interface 21
APPENDIX D: 26
Microcontroller Code to Communicate with the ESP8266 26
APPENDIX E 30
Python Code to run the Webserver 30

1. Introduction

1.1 Objective

People often forget about their laundry and in shared laundry rooms this can back up the machines and cause issues. More importantly, clothes may become damaged or get a foul smell that is hard to remove if left sitting damp in the washing machine for long periods of time [1]. This can be a costly mistake, especially for students who already live on a small budget, as one load of laundry can be worth up to $100 or more in clothing [2].

Another issue that can arise, especially in busy laundromats or coin laundry stations, is that if someone doesn't pick up their laundry then the next person can't start their load and that delay is money lost for the business or coin operated unit. Even worse, someone could forcefully remove the clothes to put their own in, which could damage the load or force the user to rewash their clothes. According to the T&L equipment company, "The average wash costs you $2.50 and the average cost to dry is $1.25. This is typically 50% of one's washing machine revenue. So, on an average day laundromat, X, has 20 machines at 4 uses a day at $2.50 per use is around $200 and another $100 for the dryers. That equates to $300 of profit for a typical day [3]." If congestion becomes an issue and the machines get backed up to three uses per day for example, then you are looking at of $240 profit on a typical day. That would cause a loss of $60 per day and lowers the more congested the laundromat becomes. Team 8's solution to this issue is to create a device that will notify the user when their laundry has completed its task, preventing congestion and backup

We believe that Laundry Alarm would be a perfect solution for everyone from students in dorms or apartments to large businesses that run multiple coin operated machines. Dorm and apartment laundry rooms are always busy, and people often forget or leave their load in the washer or dryer for long periods of time after the load has been completed. The same issue can arise in busy laundromats. Laundry Alarm helps eliminate congestion and traffic in laundry rooms and help people save time by notifying them as soon as their laundry is ready for pickup.

A device like this is currently not available on the market, except for brand-new washers and dryers that have a similar feature built-into the device, but those solutions can be very expensive, costing upwards of 1000$. The solution, however, would work on all models by simply plugging one's washer or dryer unit into the device and receiving notifications on the mobile device for a fraction of a cost.

1.2 Background

According to Lucinda Ottusch, the lead home economist at the Whirlpool Institute of Fabric Science, “If you let your wet clothes sit long enough, you'll have to deal with not only the mildew smell in your clothes, but also possible stains and patches of rotten fabric. [1]” Depending on the quality of clothes you own, this can cost you up to $100 in clothing that could be ruined [1]. This would be a very costly problem especially for students who are already on a tight budget.

Also, according to Pheena Kenny, who works with the Asthma Society of Ireland, “The moist environment of one’s washer encourages mold and mildew to release spores. This can cause problems for people with conditions like asthma. "Normally, when people breathe in these spores, their immune system helps get rid of them by coughing or sneezing. If you aren’t sensitive to mold, you may never even experience a reaction [1]”. According to the center for disease control and prevention about 1 in 12 people in the United States (about 25 million) have asthma, and the numbers are increasing every year [8], so our product has the potential to help millions of people with allergies. By making sure the user of the laundry machines does not let laundry sit and give the potential to collect mold and dust.

One existing (DIY) solution to this problem that we discovered online is a vibration sensor that is mounted on the outside of the machine. That, however, is not a good solution because calibrating the vibration sensor would be very difficult for the wide variety of washing machines on the market. Every machine would vibrate at a different frequency and magnitude, so it would be impossible to make a universal solution using vibration sensors. Vibration from nearby machines would also interfere with measurements. Another thing is that many new washing machines have advanced vibration dampening so newer, more expensive models would not work with vibration sensors at all. The device, which will measure current draw, is a universal solution that will work with any model, whether it’s 30+ years old or brand new.

1.2 High-Level Requirements

- Laundry Alert will have to operate on a wireless network and maintain a data transfer rate as described in the ESP8266 Wi-Fi chip of around 11520 baud, the amount needed to maintain a proper server connection between the microcontroller and the android application.
- The device will have to operate within a building and have a range of 150 to 200 feet from the router.
- The device will need a software delay of around 1 minute of zero current (or below threshold current) before sending the alarm to avoid sending a false alarm when the washer or dryer is between cycles or in a cycle that draws little to no current.
- The device will be an affordable solution (less than $50) to the problem we mentioned earlier.

2 Design

2.1 Physical Design

This device requires three main components to function properly: power, control, and a Wi-Fi communication module. The device will be powered from a wall outlet (120V AC) and we will implement an AC/DC converter to get two different output voltages, being 5 and 3.33 volts. The AC/DC converter will be used to power the low voltage components of the device like the microcontroller and Wi-Fi module. Another thing to consider is the device will have to both power the washer or dryer and collect current data at the at the same time using a current transformer (CT) around the hot wire. The CT will be loaded with a resistor, and the voltage across the resistor will be used to collect data for the microcontroller. After the data is analyzed, a GET packet will be sent in HTTP format to a webserver from a signal sent out from the microcontroller to the ESP8266 Wi-Fi module. It will then be displayed to the user via a SMS message sent to the user form the credentials they provided on the webserver.

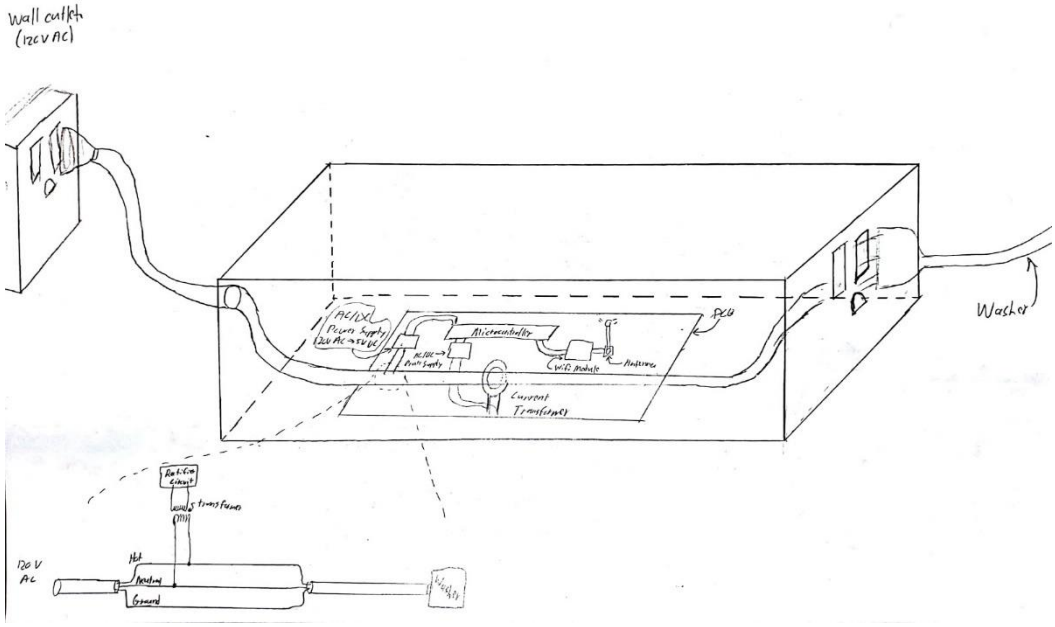


Figure 1: Physical Design

Figure 1 represents a 3D model of Laundry Alarm. The rectangular enclosure or casing is made of ABS thermoplastic. This enclosure will have an outlet on the right to plug in a washer or dryer and a 120V AC cable from a wall outlet will be led through the casing (through a 21mm cable gland) to a terminal block in the center of the device. The zoomed in diagram in *Figure 1* shows where the wire will be tapped to power the low voltage components.

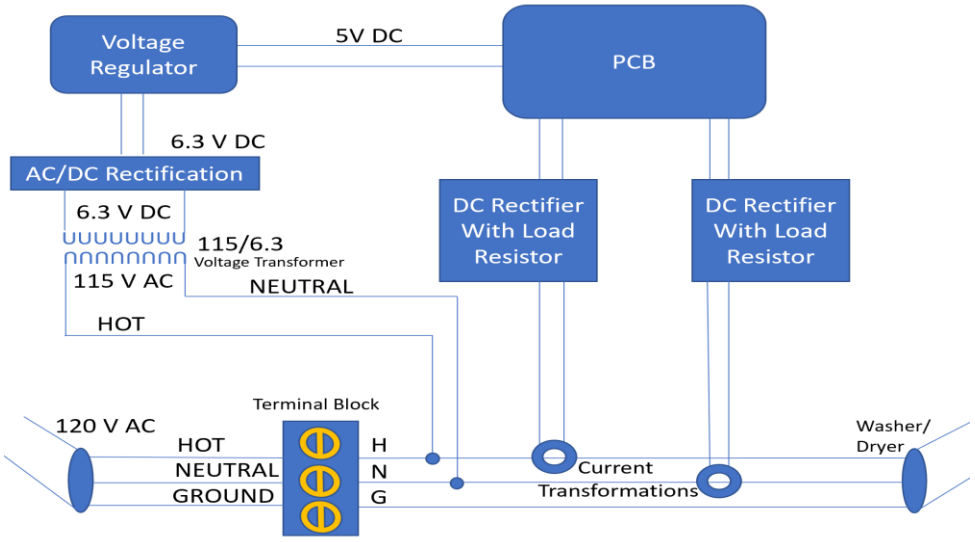


Figure 2: Wiring Diagram

Figure 2 shows the wiring diagram for the low voltage components of the device. The 120V AC wire from the outlet will be split at the end and attached to a terminal block in the center of the device. On the right, 12-gauge wires will connect the terminal block to a properly rated outlet where the washer or dry would plug in. A transformer, connected to the hot and neutral screws of the terminal block, will then be used to step down the voltage to 6.3 V AC. Then a full bridge rectifier circuit and some shunt capacitor will be used for AD/DC rectification. Finally, a voltage regulator will be used to get the desired 5V input for the microcontroller and 3.3V input for the Wi-Fi module.

2.2 PCB Design

The printed circuit board (PCB) for the Laundry Alarm contains the circuits for the power supply, microcontroller, and Wi-Fi communication modules. As shown in *Figure 3*, the voltage rectification and regulation circuit both receive the stepped down voltage from the voltage transformer which in turn provides power for the low voltage components of the device like the ATmega328P-PU and ESP8266-01 after rectification and regulation. The Current Transformer circuit takes the outputs of the CT's connected to the hot and neutral wires passing through the device and then proceeds to rectify the incoming signal and send it across a resistor load. The voltage across the resistor load is then sent to the ATmega where that data is analyzed and sent out to the ESP8266, which then sends an HTTP GET packet to the web application which determines the current status of the application. The Control and Communications circuit is responsible for collecting data from the current transformers and then relaying that data to the ESP8266 which communicates with the web application enabling the user the ability to see the current data. Finally, the crystal oscillator circuit synchronize the clock speeds with the baud rates of the ESP8266-01 to 16MHz, at 9600 baud.

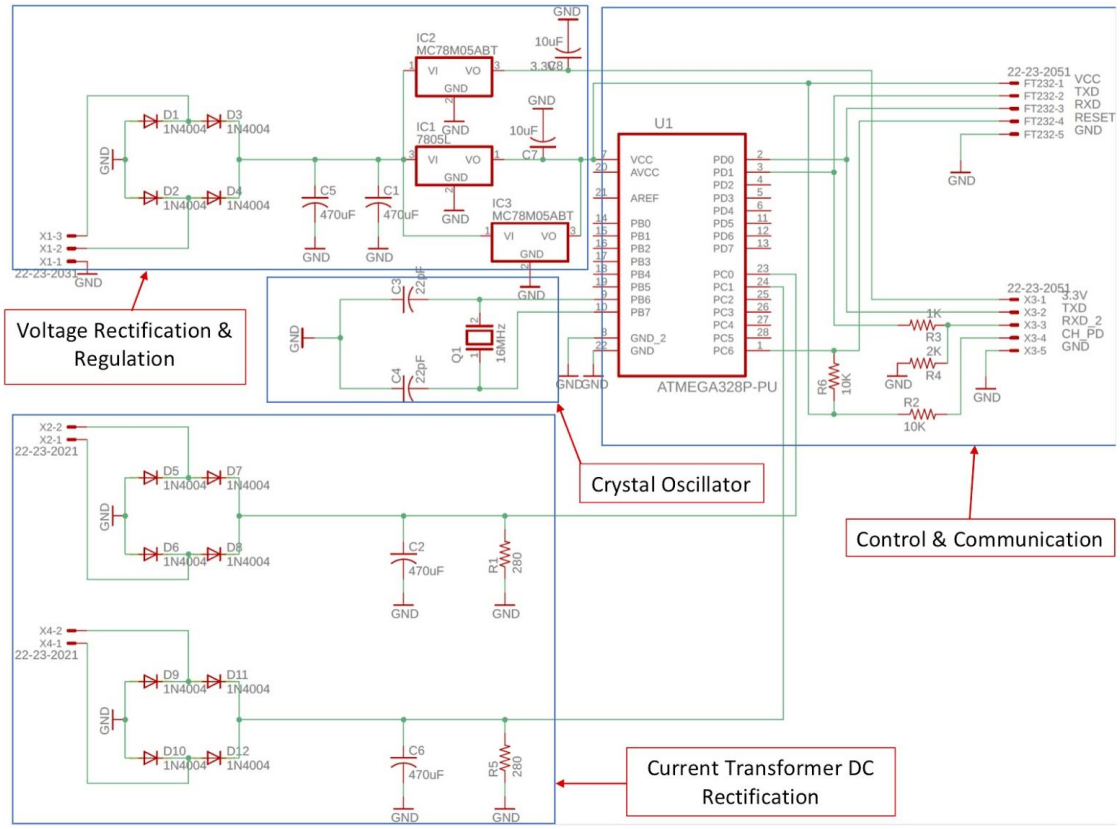


Figure 3: Circuit Design

Figure 3 represents the circuit diagram for the Laundry Alarm device. Each subsystem is blocked and labeled.

2.3 Block Diagram

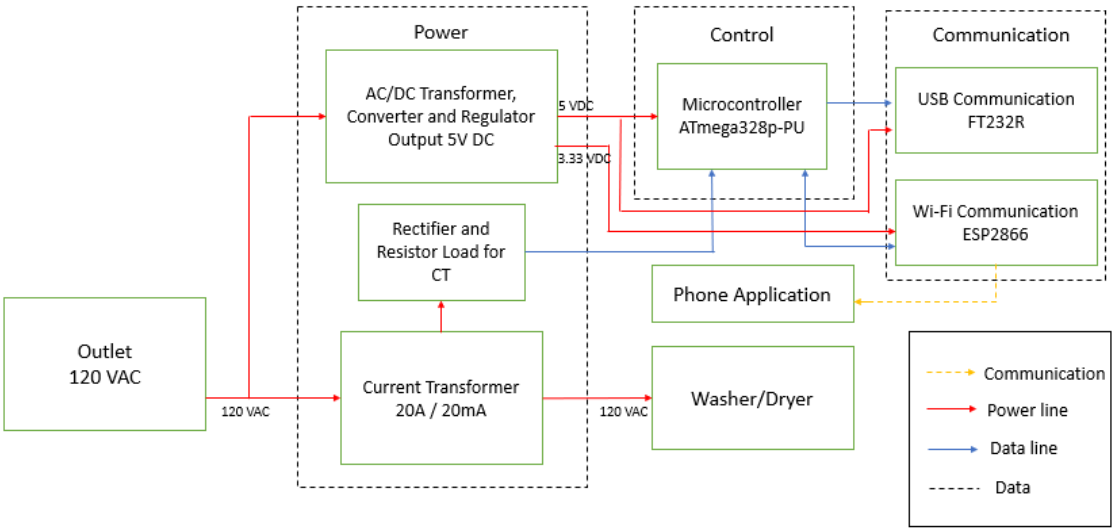


Figure 4: Block Diagram

Figure 4 represents the block diagram for the Laundry Alarm Device. This diagram shows how the washer/dryer will be powered through the device, how the low voltage components of Laundry Alarm will be powered, how the current data will be collected and processed through the microcontroller, and how the device will communicate with the web application.

2.4 Current Transformer Load Design

To calculate the load for our current transformer to output 5V at 20mA, we need to find the average current after rectifying the waveform:

$$I_{avg} = \frac{1}{T}\int_{0}^{T} I_{RMS}\sqrt{2} * sin(\theta)\, d\theta \quad \textbf{(1)}$$

$$I_{avg} = \frac{1}{\pi}\int_{0}^{\pi} 20 * 10^{-3}\sqrt{2} * sin(\theta)\, d\theta \quad \textbf{(2)}$$

$$I_{avg} = 18.006\ mA \quad \textbf{(3)}$$

$$With\ V_{out} = 5V,\ R_{load} = \frac{5}{18.006*10^{-3}} = 277.68\ \Omega \simeq 280\ \Omega \quad \textbf{(4)}$$

2.5 Control Unit

The control unit, as seen in *Figure 5,* will receive current draw data from the current transformer and that will then be analyzed and determine when to notify the user through a mobile application. If the current falls below some threshold for around a minute, the control unit will send a signal to the Wi-Fi module and then to the user via an SMS message and/or an E-mail message.

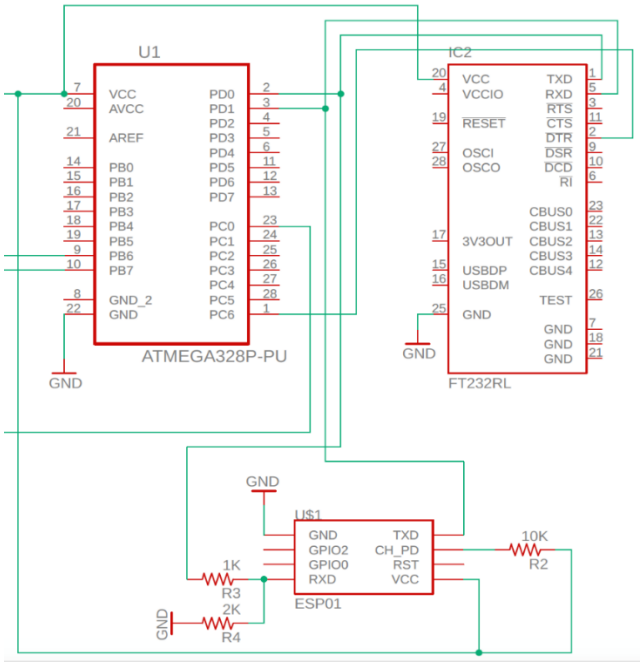


Figure 5: Control Unit

Figure 5 represents the circuit design for the control Unit

2.5.1 Microcontroller

The circuit for the microcontroller is shown below in *Figure 6.* The microcontroller's purpose is to poll for input analog values received from the PCB's neutral and hot wire. It reads in the voltage values and converts them to current values to be sent to a webserver. The microcontroller enables communication with the ESP8266 Wi-Fi module at 9600 baud. The initial setup opens a serial communication port with the ESP8266 and communicates with the module using AT Commands, a common IOT protocol used for communication with Wi-Fi, LTE, and other IOT devices.

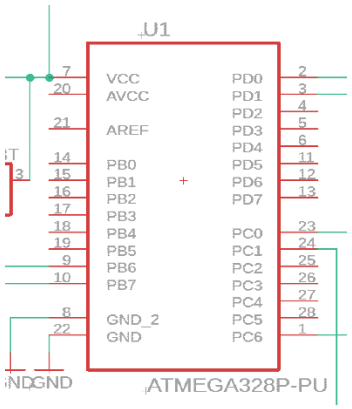


Figure 6: ATMega328P Microcontroller

Figure 6 represents the ATmega328p-pu microcontroller on the circuit diagram

2.5.2 Wi-Fi Module

At its initial startup, the ESP8266 will check if it can communicate with the microcontroller through the microcontroller sending the "AT" command, as seen in *Appendix C Figure 8*. If the ESP8266 receives the response "OK", then correct communication between the devices has been established. The microcontroller proceeds to then reset the ESP8266 and attempt to connect the ESP8266 to the internet. A process which can be seen in further detail within the code in *Appendix D*. The microcontroller proceeds to then push out HTTP GET packets to the webserver containing the specific unit's ID, the current of the device, and the current leakage calculated from the device. The ESP8266 can now send information over the network.

In order to send data to the webserver, the microcontroller must tell the ESP8266 to initialize a TCP port in order to send a GET push command for the python code to interpret and put into the database. The sequence is seen below in *Appendix C Figure 9* and can be seen in more detail in *Appendix D*.

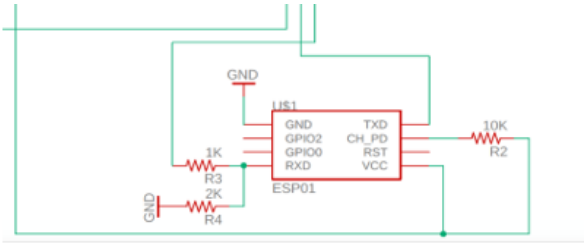


Figure 7: ESP8266 Module

In figure 8, the ESP8266 Wi-Fi module is used to transmit data using Wi-Fi to an online webserver. It receives AT commands from the microcontroller.

2.6 Software

The software for this device composes of the whole method of communication between the User Interface, the database for the User Interface and how the Microcontroller communicates between the two. The general idea can be seen blow in *Appendix C Figure 10*.

2.6.1 User Interface

The user interface (UI) is what is the application the user utilizes to view data and enter information. The user interface was generated utilizing a python script and a MYSQL database which stores the information received from the microcontroller on the PCB. Each laundry alarm device has a unique identifier which acts as the device's name so that the database and web UI can distinguish which device it needs to display. The overall flow chart for the program is described in *Appendix C Figure 11*. The program itself can be found in *Appendix E.* The Web UI waits for a current input larger than an initial threshold, then it triggers a start response from the application and it begins reading current values once every 30 seconds. The application sends both a finish E-mail and SMS message to the user once three values below the starting threshold have been processed. This is important because generally

Washer/Dryer machines will have momentary stops in current as it is going through the cycle, having three values below the threshold instead of one eliminates possibility of a false positive, or sending a message when the machine has not yet completed.

To access the application. The user will scan a QR code on any cellular device that has the capability to do so. It will redirect the user to the check session tab as seen in *Appendix C Figure 12* and will display the page as seen in *Appendix C Figure 13*. The user will then be taken to a screen in which they can enter their phone number and/or email to receive the type of notification(s) they would prefer. This is seen in *Appendix C Figure 13.* After entering the information for the notifications, if the user has opted to receive notifications from either text, E-mail, or both, they will receive the following notifications in *Appendix C Figure 14* and will be prompted to go to the active session page where the user's current session information can be viewed. The Application will redirect the user to a page with a link that when pressed will send the user to the page to check the user's recent activity which checks for 3 values below the threshold in order to send the finish notification to the user as seen in *Appendix C Figure 15* and *Appendix C Figure 16*. Upon receiving the third value below the threshold, the web application will send out a completed message as seen in *Appendix C Figure 17.*

3 Design Verification

The verification of the design is based on the requirements for each individual sub-system as seen in *Appendix A.* Each module was tested individually to ensure that each requirement is met and that the subsystems can work together to make a final working product.

3.1 Power

3.1.1 AC/DC Power Supply

To ensure proper operation of the voltage regulator, input voltage to regulator must be between 5 and 12V. The power supply will need to provide a rectified and filtered output with the minimum ripple voltage exceeding 5V and the maximum ripple voltage below 12V. To verify this, we probed the output of the rectifier circuit to make sure the voltage ripple is within the specifications of the voltage regulators. The value probed at the output was 7.2V with less than 10 mV of ripple. The table for this is referenced in *Appendix A Table 2.*

3.1.2 Voltage Regulation

The regulator circuit must step down 7.2 V DC from rectifier circuit to 5V ± 5% for the microcontroller. The other regulator circuit must step down 7.2 V DC from rectifier circuit to 3.33V ± 5% for the ESP2866. To verify this, we used a multimeter to read the voltage across the output terminals of the regulators to check if the voltage was within the specified range. The results were precisely 5V out of the 5V regulator and precisely 3.33V out of the 3.33V regulator. These regulators were very accurate for the small power draw. The table for the verifications is listed in *Appendix A Table 2*

3.1.3 Current Transformers

The voltage across the current transformer load must be below the absolute maximum analog input of 6V. As explained in Tolerance section 4.2, the voltage across the load is within this range with maximum component tolerance error if the load stays below the maximum CT input of 20A. The largest voltage we logged was 2.95, which is well within the tolerance of 6V. The table for the verifications is listed in *Appendix A Table 2.*

3.1.4 Leakage Current

The Laundry Alarm device will be able to detect the difference in current between the hot and neutral wire. Similar to a standard GCFI, if we detect a current greater than 10-20mA, the user will be notified that there may be a potential problem with their Washer/Dryer machine. The table for the verifications for the leakage current can be found in *Appendix A Table 7.*

3.2 Control Unit

3.2.1 Microcontroller

The ATmega32P-PU microcontroller must receive current data from the current transformer. It also must be able to communicate with UART at rates > 50 kbps, the fastest speed the microcontroller can run without significant data loss and it must be able to run at a voltage of 5.0V +/- 5%. The verification for this was accomplished by running the microcontroller via a serial monitor and testing the responses back from the ESP8266 module to see if they were outputting correctly with no errors. The verifications can be found in *Appendix A Table 3.*

3.2.2 Wi-Fi Module

For the Wi-Fi Module, the verifications can be seen in *Appendix A Table 4*. The Wi-Fi module was required to run at 3.3 volts which was tested by probing the device and ensuring the device turned on properly when powered. There was a required transmission rate of 125 feet which was tested by first setting up a proper internet connect through the ESP8266 by using AT commands to initialize the network with the command “AT+CWJAP = “WI-FI network”, “Password””. After an OK response was read the ESP8266 was successfully connected to the internet. The ESP8266 was also required to maintain a data transfer rate of around 14 kBps, this was accomplished by monitoring the rate at which successful HTTP GET post commands were sent to the MYSQL database. Walking around with the device enabled testing capabilities of the transmission rate while testing it on different wireless networks.

3.2.3 Android Application

For the android application, as seen in Table 5 in Appendix A, it was supposed to receive a signal from the microcontroller through the ESP8266 to send the user a notification upon the completion of the laundry cycle. This was tested by creating an application through Android studio which utilized a backend server through Firebase. The original application layout can be seen in *Appendix C Figure 18*, which shows the original android application layout and received notification before switching over to the web application. The app ended up being unsuccessful, as the back-end server was new and kept rearranging the security protocols, deprecating crucial functions that were necessary to send out messages to the user. A more appropriate web application was created in its stead. The web application successfully sent out both an E-mail and SMS text notifications to the user, providing an alert to notify the user when they had authenticated a session and notifying the user when the laundry cycle had completed. This web-application ended up being more successful than the original idea of the android application because it was now geared towards all users and not just users that had an android phone.

4. Costs

The cost analysis portion in the following chapters will show in detail the cost of every component that went into the Laundry alarm, the final cost to produce one unit, and the total cost of research and development and labor costs.

4.1 Parts

Appendix B Table 8 represents the costs of everything utilized to create the project including quantity of parts used for one device, retail cost of each part, cost of parts for research & development, and cost of parts to make one device. The total retail cost of the parts is $53.25. The total cost for all parts in research and development is $94.51. The cost to build a working device is $66.73.

4.2 Labor

Group Member	Hourly Rate ($/hour)	Time Invested (hours)	x2.5	Total Cost ($)
Michelle Lamblin	40	100	x2.5	10,000
Marcin Horwat	35	100	x2.5	8,750
Thomas Mayer	35	100	x2.5	8,750
Total Cost				$27,500

Table 1: Labor Costs

5. Conclusion

5.1 Accomplishments

Overall, the device is fully functional, and all subsystems work and they work together. The device can successfully monitor current and leakage current of any washer or dryer plugged into the device and provide that data to the user. The device also sends notifications to the user by text or email successfully to notify the user when their wash or dry cycle is complete once the laundry cycle of either the washer or dryer has completed.

5.2 Uncertainties

One small issue with the device is that the crystal oscillator circuit was not built properly and does not always work. The main issue with the current circuit is that the crystal is too far away from the microcontroller and that it is a through hole component. The long path/trace length and through hole adds a lot of unnecessary impedance to the circuit, which really distorts the signal produced by the 16MHz crystal. Another issue with the current design is that there is no ground plane on the top layer of the PCB, so leakage and noise from other nearby circuits are interfering with the crystal oscillator circuit, causing it to malfunction.

To fix this we would have to use a pad mounted crystal instead of a through hole component to help eliminated the added impedance. If a through hole is the only option available, then the solution for that would be to implement a guard track on the bottom layer of the board parallel to the track on the top layer. This will allow for a better connection between the microcontroller and the crystal oscillator. To eliminate leakage and noise from other nearby circuits we would have to implement ground guard or ground plane around the crystal oscillator circuit to essentially fence it off from other circuits. *Figure 19* **below** shows the old design on the left and a new and improved design on the right.

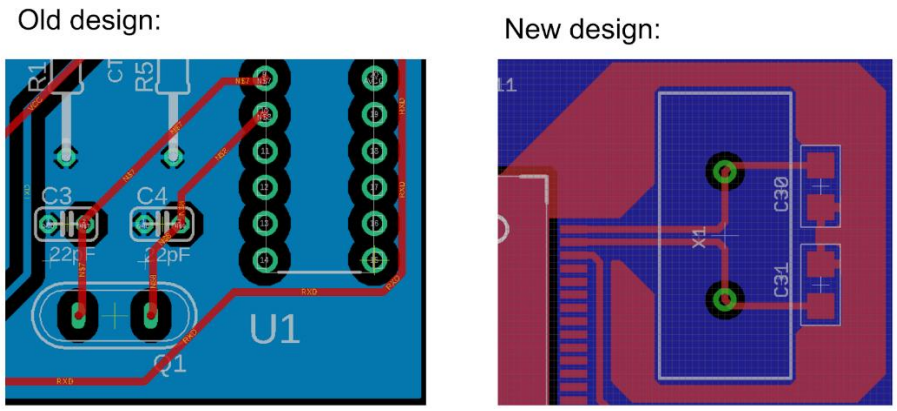


Figure 19: Physical Design

Figure 19 shows the current crystal oscillator circuit, that does not work properly, and a new and improved design that will eliminate all the current problems with the circuit. The new design has the

crystal much closer to the microcontroller compared to the old design and it also has a ground plane around it to protect the circuit from leakage and noise coming from nearby circuits.

5.3 Ethical considerations

The primary concern with regards to project safety is the 120VAC outlet power we will be working with. It is essential that the process of lowering the DC voltage to a limit where the PCB is safe is ensured. The live 120VAC power will cause concern and we will need to take the proper caution when working with it. As per the lab safety standards, we will make sure to check for any damage to the outlet/wires and will not bring food/drinks into the lab to avoid potentially getting liquids on any electronics and prevent damage to the electronics.

Along with taking extreme caution while using 120VAC, we will also follow Section 6 of the IEEE code of Ethics[7] while working with voltage converters and regulators and to keep true to Section 7 of the IEEE code of Ethics[7], making sure the work to create and surround the creation of the project Laundry Alarm is compliant with ethical designs and sustainable development practices when working with 120VAC.

In terms of malicious interception of signals, the system could potentially be hacked, we intend to avoid this by creating a user and password authentication requirement when we enable Wi-Fi on the network, so a potential malicious user would not gain an easy access way into a user's personal Wi-Fi. This will be done through proper API authentication ensuring a user will have to enter a username and password which will be encoded within the program to ensure someone who obtains the code cannot have access to the user's Wi-Fi login information.

5.4 Future work

Although the Laundry Alarm was successful for the most part, there is still room for improvement or additional features in the future. Some possible additional features include:

- Scaling to laundromat use with multiple devices
- Applying current sense device to other appliances or devices
- Integrating GFCI or leakage current protection into the device
- 220 V Integration
- two outlets for monitoring washer and dryer at the same time (2*120V or 1*120V and 1*240V)
- Making an Android/iPhone application

References

[1] Aman, Melanie. “How Long Can Wet Clothes Stay in the Washer?” *Woman's World*, Woman's World, 9 Apr. 2018, www.womansworld.com/posts/how-long-can-wet-clothes-stay-in-the-washer-127358.

[2] Collings, Kat. “How Much You Should Really Be Spending on Clothing Every Month.” *Who What Wear*, Who What Wear, 1 Mar. 2018, www.whowhatwear.com/monthly-shopping-clothing-budget-guide-2014.

[3] “Commercial Coin Operated Laundry Equipment NC | How Much Money Ca.” *T&L Equipment Company | Commercial Laundry Equipment Experts*, 17 Mar. 2017, www.washcycle.com/much-money-can-really-make-owning-laundromat/.

[4] “Controlling Electrical Hazards.” *Occupational Safety and Health Administration*, www.osha.gov/Publications/3075.html.

[5] “Full Wave Rectifier and Bridge Rectifier Theory.” *Basic Electronics Tutorials*, 24 Feb. 2018, www.electronics-tutorials.ws/diode/diode_6.html.

[6] “IEEE Code of Ethics.” *IEEE - Advancing Technology for Humanity*, www.ieee.org/about/corporate/governance/p7-8.html.

[7] “The Code Affirms an Obligation of Computing Professionals to Use Their Skills for the Benefit of Society.” *Association for Computing Machinery*, www.acm.org/code-of-ethics.

[8] “Vital Signs.” *Centers for Disease Control and Prevention*, Centers for Disease Control and Prevention, 3 May 2011, www.cdc.gov/vitalsigns/asthma/index.html.

[9] Fluke. “Leakage Current Measurement Basics.” Leakage Current Measurement Basics | Fluke, Fluke Corporation, 9 May 2013, www.fluke.com/en-us/learn/blog/clamps/leakage-current-measurement-basics.

Appendix A

Requirement and Verification Table

Requirements:	**Verifications:**
Power: 1. To ensure proper operation of the voltage regulator, input voltage to regulator must be between 5 and 12V. The power supply rectified and filtered output is rated to be within this range with maximum component tolerance error.	A. Scope the output and make sure the output voltage ripple is within specification of the voltage regulator.
2. Regulator circuit must step down 6.3 V DC from rectifier circuit to 5V ± 5% for the microcontroller. Regulator circuit must step down 6.3 V DC from rectifier circuit to 3.33V ± 5% for the ESP2866	A. Use voltmeter across the output terminals of the voltage regulator to check if voltage is within the specified range B. Use voltmeter across output terminals of the voltage regulator to check if voltage is within the specified range.
3. The voltage across the current transformer load must be below the absolute maximum analog input of 6V. As explained in Tolerance section 4.2, the voltage across the load is within this range with maximum component tolerance error if the load stays below the maximum CT input of 20A.	A. Create a test circuit with 20 A on a wire to measure the voltage across the load of the CT and ensure that it below 6V. This can be measured with a voltmeter.

Table 2: Power Requirements and Verification

Requirements:	**Verifications:**
Microcontroller: 1. Must receive current data from the current transformer	A. Create a simple program for the microcontroller that can receive that data obtained from the current transformer. B. The program will read the data from the analog pins of the microcontroller and display the values to the computer.

	C. We will test the output pins with an oscilloscope to determine if the signals are being transmitted properly.
2. The microcontroller must be able to communicate with UART at rates > 50 kbps, the fastest speed the microcontroller can run without significant data loss.	A. Establish a bridged communication between the microcontroller and a form of terminal emulator like PUTTY. B. Set the Baud rate to be 9600 bps or (50 kbd) via the SERCFG command which allows baud rate configuration parameters to be set over PUTTY through the micro-USB cable.
3. Must be able to run at a voltage of 5.0V +/- 5%	A. Probe the input of the microcontroller with an oscilloscope to ensure it can run at the desired voltage range

Table 3: Microcontroller Requirements and Verification

Requirements:	**Verifications:**
Wi-Fi Module: 1. Must be able to run at a voltage of 3.0V +/- 5%	A. Move the voltage over the range of 3 +/- 3 volts and see if the AT command AT returns the response OK from the ESP8266 from the microcontroller.
2. Must have a transmission rate of around 125 +/- 25 feet.	A. module over the microcontroller and type the commands AT to make sure it is working properly. B. The command AT should return OK. C. The command AT+CWMODE=1 and AT+CWTAP=" Wi-Fi network", "password" must return OK
3. Must allow for a data transfer rate of around 14 +/- 5% kBps	A. Open a web browser and enter the IP of the Wi-Fi module to receive data and open a putty session and set the proper baud rate using the SERCFG command and establish a server connection with the Wi-Fi module. B. Now, type in any length of characters > 50 and it should be echoed on the web page through HTTP.

Table 4: Wi-Fi Module Requirements and Verification

Requirements:	Verifications:
Android application: 1. Must be able to receive a signal from the microcontroller through temboo at the microcontrollers rate of 50 kbps +/- 10%	A. Create a test application on an android development studio see figure 3.9. B. Create an account on parse.com and import all the packages. C. Compile and test to see if a popup notification appears correctly through the android app on the emulated environment.
2. Must be able to send a popup notification within 5 +/- 1 second to an emulated android environment.	A. The notification must be sent within 4-5 seconds of the signal being received.

Table 5: Android Application Requirements and Verification

Requirements:	Verifications:
Case: 1. Design should fit all components inside a 6.1” x 4.6” x 2.37” case	A. Check dimensions of all components and ensure that they will fit inside the case with the specified dimensions. Case must seal and screw on securely.

Table 6: Case Requirements and Verification

Requirements:	Verifications:
Leakage Current Detection: 1. Detect the difference in current between the hot and neutral wire. Like a standard GCFI, if we detect a current greater than 10-20mA, the user will be notified that there is an issue with their machine	A. Load the output of the device to operating current. While the device is loaded, switch on a power resistor between the hot and ground to simulate a leakage current. The power resistor will be valued to draw just over 20mA. The device should then send a warning signal to the user.

Table 7: Current Leakage Detection Requirements and Verification

Appendix B

Cost Analysis Table

Part	Manufacturer	Quantity	Retail Cost ($)	Bulk Purchase Cost ($)	Actual Cost ($)

Bud Industries enclosure (Case)	ECE Supply Shop	1	**$9.90**	**$9.90**	**$9.90**
Current Transformer(20A/20mA)	Digikey	2	**$10.22**	**$20.44**	**$20.24**
Voltage Transformer (115/6.3)	Digikey	1	**$8.99**	**$10**	**$8.99**
Voltage Regulator (5V)	Digikey	1	**$1.53**	**$7.65**	**$1.53**
Microcontroller (ATMEGA328P-PU)	Digikey	1	**$2.14**	**$6.42**	**$2.14**
Wi-Fi Module (Mouser) (ESP8266)	Digikey	1	**$6.95**	**$13.9**	**$6.95**
Capacitor (470µF,25V)	Digikey	4	**$0.68**	**$2.72**	**$2.72**
Capacitor (22 pF)	Digikey	2	**$0.40**	**$1.60**	**$0.80**
Diode (1N4004)	Digikey	12	**$0.11**	**$1.32**	**$0.88**
Outlet(103.124mm x 33.274mm x 26.924)	ECE Supply Shop	1	**$6.68**	**$6.68**	**$6.68**
3-Prong Power Cord with Open Wiring - 6 ft.	ECE Supply Shop	1	**$3.29**	**$6.58**	**$3.29**
280 Ω Resistor	Digikey	2	**$0.15**	**$0.75**	**$0.30**
1 KΩ Resistor	Digikey	1	**$0.10**	**$0.50**	**$0.10**
2 KΩ Resistor	Digikey	1	**$0.29**	**$1.45**	**$0.29**
10 KΩ Resistor	Digikey	1	**$0.10**	**$0.50**	**$0.20**
16MHz Crystal	Digikey	1	**$0.66**	**$1.98**	**$0.66**
SO Cord connector	Digikey	1	**$1.06**	**$2.12**	**$1.06**
Total			**$53.25**	**$94.51**	**$66.73**

Table 8: Cost Parts Analysis

Appendix C:

Photo References for Software and User Interface

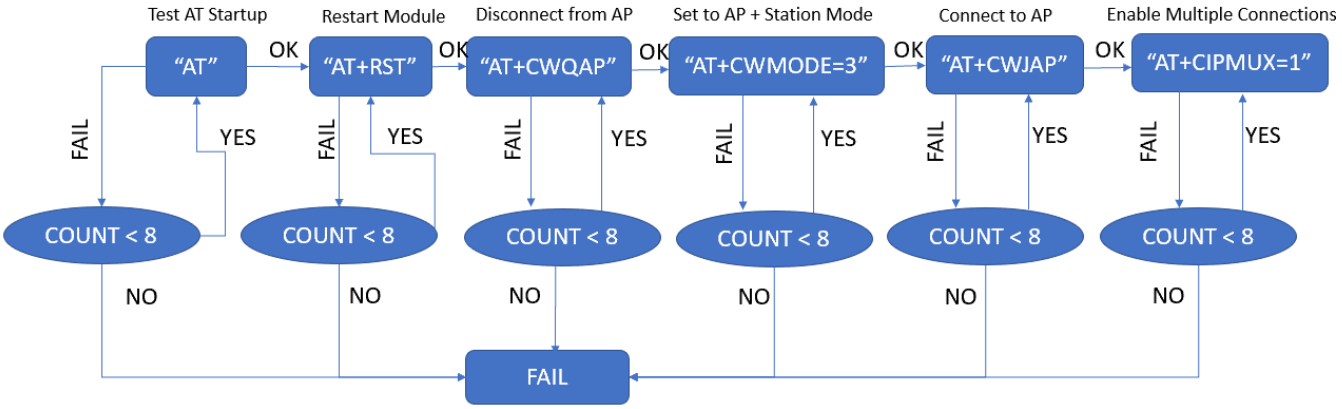


Figure 8: AT Startup Sequence

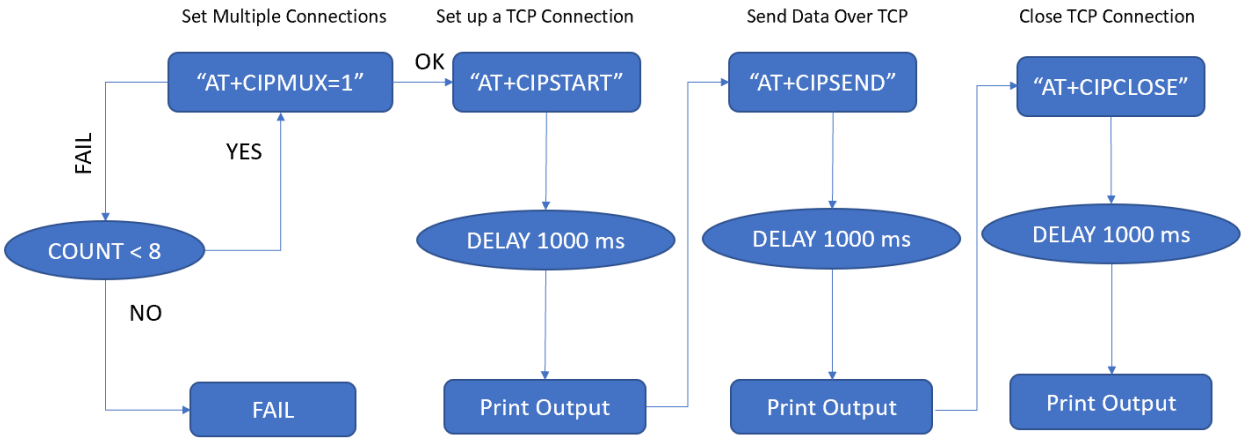


Figure 9: AT Send Data Sequence



Figure 10: High Level Software Communication

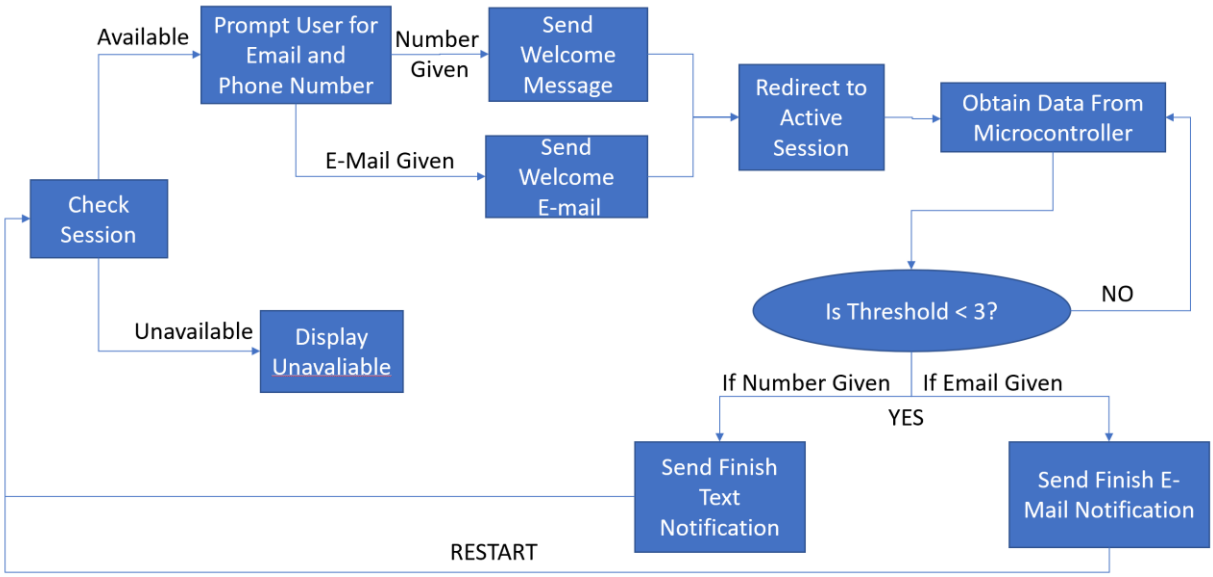


Figure 11: UI Flow Diagram



Figure 12: UI Welcome Page and QR Code

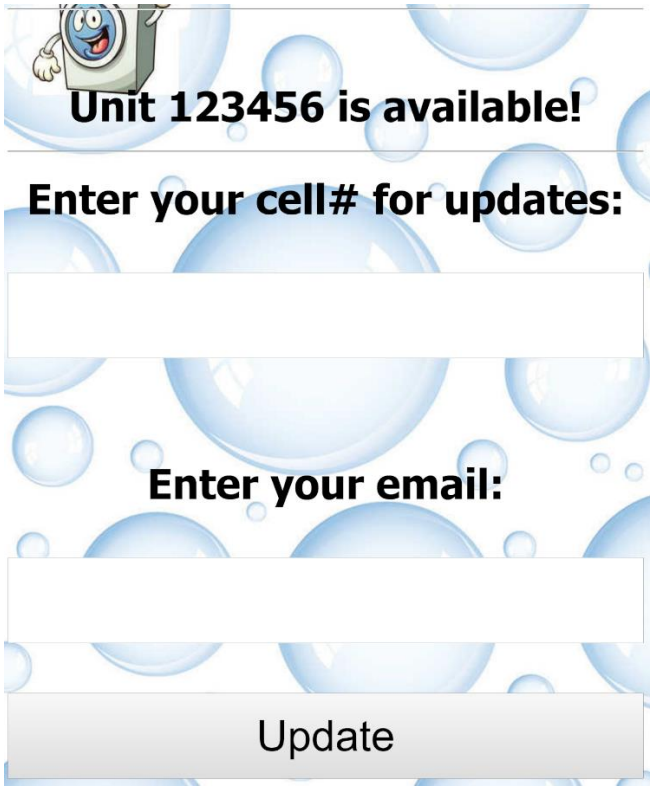


Figure 13: UI User Input Page

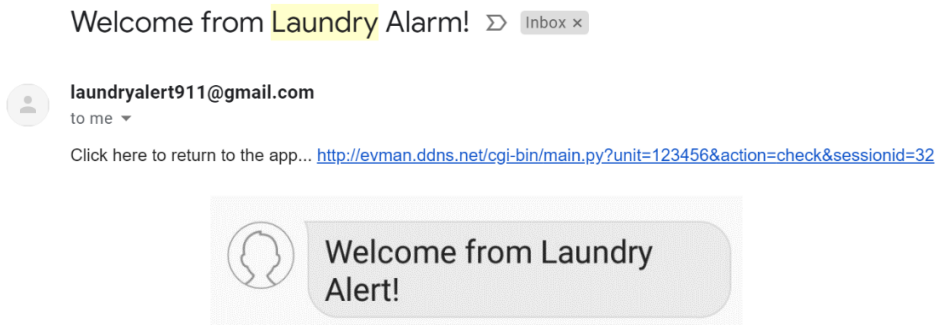


Figure 14: UI Initial E-mail and Text Notifications



Figure 15: UI Redirecting Page



Figure 16: UI Page to view Data Values

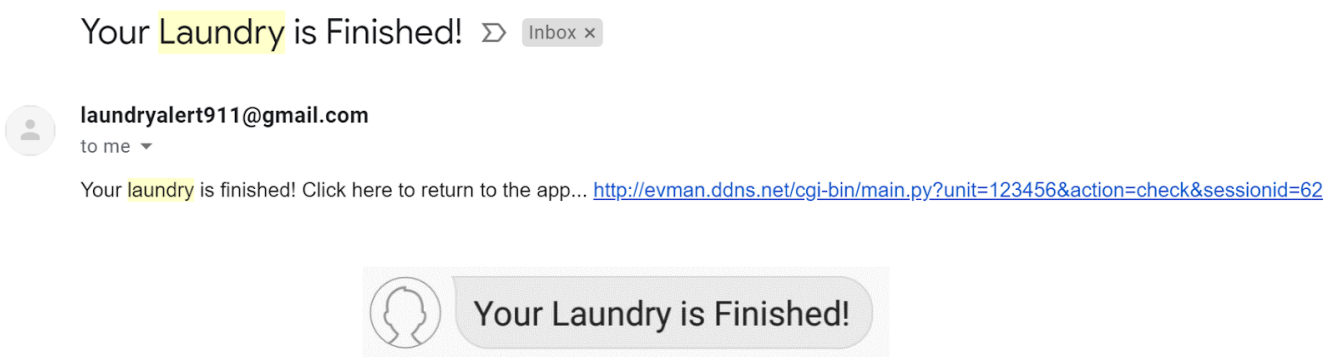


Figure 17: UI Finished Email and Text Messages

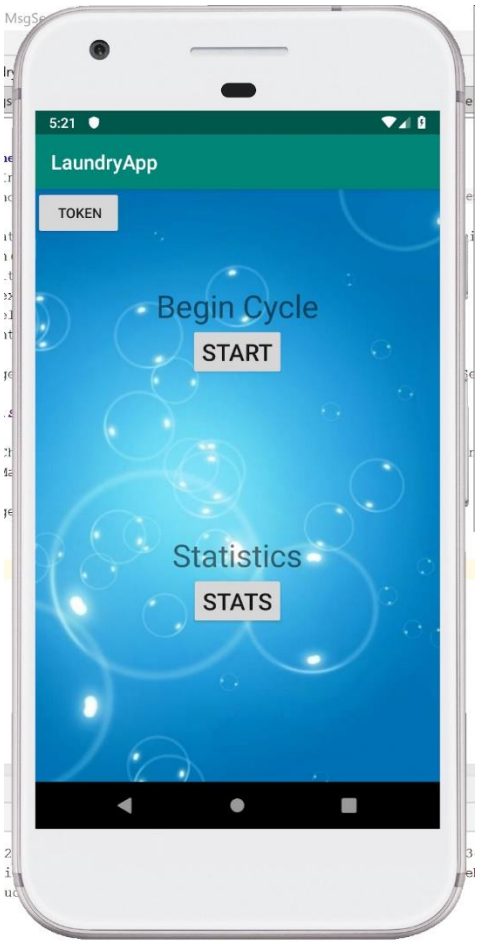


Figure 18: Original Android Application

Appendix D:

Microcontroller Code to Communicate with the ESP8266

```
/*Read Analog Data from Atmega328P
 * PC0 - Hot wire current read
 * PC1 - Neutral wire current read
 */
#include<math.h>
#include<SoftwareSerial.h>
SoftwareSerial client(2,3); //RX, TX
static const unsigned HotIn = A0;
static const unsigned NeutIn = A1;
int i=0,k=0;
int x=0;
boolean No_IP=false;
String IP="";
char temp1='0';
// Strings below not used - reference only
String server = ":<YOUR DOMAIN>';
String uri = "/cgi-bin/main.py";
String id="123456";   //22
int current = 0;
int neutral = 0;
int leakage = 0;
double current1 = 0.0;
double neutral1 = 0.0;
double leakage1 = 0.0;
int t = 0;
String device = "123456";

//This function sends the AT command to the ESP8266 and checks for the OK from the
ESP8266 signifying the command was successful
void run_cmd_ok(String cmd, int t){
       int temp=0,i=0;
       while(1){
              Serial.println(cmd);
              client.println(cmd);
              while(client.available())  {
                     if(client.find("OK"))
                            i=8;
              }
              delay(t);
              if(i>5)
              break;
              i++;
       }
       if(i==8)
              Serial.println("OK");
       else
              Serial.println("Error");
}
```

```
//This function obtains the current IP address from the ESP8266
void get_ip(){
	IP="";
	char ch=0;
	while(1){
		client.println("AT+CIFSR");
		while(client.available()>0){
			if(client.find("STAIP,"){
				delay(1000);
				Serial.print("IP Address:");
				while(client.available()>0 {
					ch=client.read();
					if(ch=='+')
						break;
					IP+=ch;
				}
			}
			if(ch=='+')
				break;
		}
		if(ch=='+')
		break;
		delay(1000);
	}
	Serial.print(IP);
	Serial.print("Port:");
	Serial.println(80);
}

//This function initializes the communication between the ESP8266 and microcontroller
and between the ESP8266 and router
void wifi_init(){
	run_cmd_ok("AT",500);
		run_cmd_ok("AT+RST",1000);
	run_cmd_ok("AT+CWQAP",4000);
	run_cmd_ok("AT+CWMODE=3",500);
	Serial.println("Connecting Wifi....");
		//provide your Wi-Fi username and password here;

	run_cmd_ok("AT+CWJAP=\"<ROUTER_NAME>\",\"<ROUTER_PASSWORD> \"",7000);
	get_ip();
	run_cmd_ok("AT+CIPMUX=1",500);

}

//This function prints the client's response to the AT+Command being sent out
void print_resp(){
	delay(t);
	while(client.available()){
		String tempResp=client.readStringUntil('\n');
		Serial.println(tempResp);
	}
}
```

```
//This function establishes communication with the ESP8266 and the ATmega328P, setting
both baud rates to 9600
void setup()  {
       Serial.begin(9600);
       client.begin(9600);
       wifi_init();
       //set up analog pins
       pinMode(HotIn, INPUT);
       pinMode(NeutIn, INPUT);
}

//This function loops over and obtains the analog voltage values and converts them to
current values
void loop(){
       current = analogRead(A0);
       neutral = analogRead(A1);
       Serial.print("Original Current: ");
       Serial.println(current);
       Serial.print("Original neutral: ");
       Serial.println(neutral);

       current1 = double(current);
       neutral1 = double(neutral);
       current1 = current1 * 5;
       neutral1 = neutral1* 5;
       current1 = current1/1023;
       neutral1 = neutral1/1023;
       current1 = current1/230;
       neutral1 = neutral1/230;
       current1 = current1*1000;
       neutral1 = neutral1*1000;
       Serial.print("Actual Current Value:");
       Serial.println(current1);
       Serial.print("Actual Neutral Value:");
       Serial.println(neutral1);
       if(current1 > 0.1){
              current1= current1 + 0.45;
              current1 = current1*1.2;
              Serial.print("After Calculation Current1:");
              Serial.println(current1);
              current1 = current1/1000;
              current1 = current1*230;
              current1 = current1*1023;
              current1 = current1/5;
              current = round(current1);

       }
       if(neutral1 > 0.1){
              neutral1 = neutral1 + 0.45;
              neutral1= neutral1*1.2;
              Serial.print("After Calcualation Neutral1:");
              Serial.println(neutral1);
              neutral1 = neutral1/1000;
              neutral1 = neutral1*230;
              neutral1 = neutral1*1023;
```

```
            neutral1 = neutral1/5;
            neutral = round(neutral1);

      }
      if(current1>neutral1)
            leakage1 = current1-neutral1;
      else
            leakage1 = neutral1-current1;
//recreate A/DC values to be passed into database
      if (current > neutral)
            leakage = current - neutral;
      else
            leakage = neutral - current;
      Serial.print("Analog Current:");
      Serial.println(current);
      Serial.print("Neutral Current:");
      Serial.println(neutral);
      Serial.print("current hot:");
      Serial.println(current1);
      Serial.print("neutral:");
      Serial.println(neutral1);
      Serial.print("leakage:");
      Serial.println(leakage1);
      k=0;
      Send();
      delay(18000);
      }
      //This function sends the HTTP GET command through the TCP port 8 on the
      webserver

      void Send()   {
            Serial.println("Got into send");
            webpage = "GET /cgi-bin/main.py?action=device&unit=" + String(device) +
"&current=" + String(current) + "&leakage="
      + String(leakage)+ " HTTP/1.1\r\nUser-Agent: curl/7.37.0\r\nHOST: <YOUR DOMAIN
NAME>\r\nAccept: */*\r\n\r\n";
            delay(2000);
            run_cmd_ok("AT+CIPMUX=1",1000);
            client.println("AT+CIPSTART=4,\"TCP\",\"<YOUR DOMAIN NAME>\",80");
            delay(1000);
            t=500;
            print_resp();
            client.println("AT+CIPSEND=4,"+String(webpage.length()));
            delay(500);
            client.print(webpage);
            delay(3000);
            client.println("");
            print_resp();
            if(client.available()) {
                  Serial.write(client.read());
            }
            client.println("AT+CIPCLOSE=4");
            String tempResp=client.readString();
            Serial.println(tempResp);
      }
```

Appendix E

Python Code to run the Webserver

```
#!/usr/bin/env python3

# Laundry Alarm main procedure
import pymysql
import cgi
import requests
import os
from datetime import datetime
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart
import math
import googlevoice

threshold = 50
fnt = '<font face=tahoma size=10>'
fnts = '<font face=tahoma size=5>'

form = cgi.FieldStorage()
action = form.getfirst("action","")
unitid = form.getfirst("unit","")
current = form.getfirst("current","")
leakage = form.getfirst("leakage","")
cellno = form.getfirst("cellno","")
email = form.getfirst("email","")
sessionid = form.getfirst("sessionid","")

print ("Content-Type: text/html\r\n")
if action != "device":
    print("\r\n")
    print("<html><head>" + fnt + "<center><b><i><Laundry Alert</center></i></head>")
if action != "device":
    print('<body background="/bubback.jpg"><form action=main.py method=post>')
cnxn = pymysql.connect(host='127.0.0.1', port=3306, user='root', password='<YOUR
PASSWORD>', db='laundry')
crsr = cnxn.cursor()

if action != "device":
    print('<font color=blue><b><i>Laundry Alert</b></i></font><hr><br>')

now = datetime.now()
formatted_date = now.strftime('%Y-%m-%d %H:%M:%S')

# action = device is the device sending data updates - need to evaluate whether the
last X subsequent are below threshold value

if action == "device":
    # Insert device data
```

```
    crsr.execute('insert into unitdata (unitid, timestamp, currentdraw, currentleak)
values (%s, %s, %s, %s)', (unitid, formatted_date, current, leakage))
    # Now update active sessions if there are any

    crsr.execute("""update session set lastactivity = %s where unitid = %s and
sessionactive = 1""",(formatted_date,int(unitid)))
    # Now update unit record
    crsr.execute("""update unit set lastdeviceinfo = %s where unitid =
%s""",(formatted_date,int(unitid)))
    print("OK")

    # initialize done to true
    isdone = 1
    idx = 0
    crsr.execute("""select timestamp, currentdraw, currentleak from unitdata where
unitid = %s order by timestamp desc limit 3""",(unitid))
    rows = crsr.fetchall()

    for row in rows:
        idx = idx + 1
        if not(row is None) and idx <= 3:
            ts = row[0]
            cd = row[1]
            cl = row[2]
            if cd > threshold:
                isdone = 0

    # need to check to see if there has been at least one unitdata record above
threshold since sessino start to make sure a valid session
    # ie - can't end session if it never started....
    if isdone == 1:
        crsr.execute("""select max(ud.currentdraw) from unitdata ud join session s on
s.unitid = ud.unitid and s.sessionactive = 1 and ud.timestamp >= s.sessionstart where
ud.unitid = %s""",(unitid))
        maxcurr = crsr.fetchone()[0]
        if maxcurr <= threshold:
            isdone = 0
    if isdone == 1:

        # get sessionid for notification to user
        crsr.execute("""select sessionid, cellno, email from session where unitid = %s
and sessionactive = 1""",(int(unitid)))
        rows = crsr.fetchall()
        for row in rows:
            if not (row is None):
                sessid = row[0]
                cellno = row[1]
                email = row[2]
        crsr.execute("""update session set sessionactive = 0 where unitid = %s and
sessionactive = 1""",(int(unitid)))
        # send email
        if len(email) > 5:
            msg = MIMEMultipart()
            msg['Subject'] = 'Your Laundry is Finished!'
```

```
            msg['From'] = '<YOUR EMAIL>'
            msg['To'] = email
            body = MIMEMultipart('alternative')
            html = 'Your laundry is finished!  Click here to return to the app...
'<YOUR WEBSERVER>' + unitid + '&action=check&sessionid=' + str(math.floor(sessid))
            body.attach(MIMEText(html,'html' ))
            msg.attach(body)
            s = smtplib.SMTP('localhost')
            s.send_message(msg)
            s.quit()
        if len(cellno) >= 10:
            voice = googlevoice.Voice()
            voice.login(email='<YOUR EMAIL>',passwd='<YOUR PASSWORD>')
            voice.send_sms(cellno, 'Your Laundry is Finished!')
# Update is creating a new session after entering email & phone information
if action == "Update":
    # Check for a cell# or email - if neither, ask for one
    if cellno == "" and email == "":
        print("<b><i><font color=red>Either an Email or Cell# is required")
        action = ""
    else:
        # Now grab last session id and create session
        sql = "select max(sessionid) as sessionid from session"
        crsr.execute(sql)
        rows = crsr.fetchone()
        sessid = int(0)
        for row in rows:
            sessid = row
            if sessid is None:
                sessid = 0
            else:
                sessid = sessid + 1

        # create new session
        crsr.execute('insert into session (sessionid, cellno, ipaddress, email,
unitid, sessionstart, lastactivity, sessionactive) values
(%s,%s,%s,%s,%s,%s,%s,%s)',(sessid,cellno,cgi.escape(os.environ["REMOTE_ADDR"]),email,
int(unitid),formatted_date,formatted_date,1))
        print("Thank you, you will be notified when cycle is completed.")
        print("<br><a href='<YOUR WEBSERVER>'unit=" + unitid +
"&action=check&sessionid=" + str(math.floor(sessid)) + ">Click here for status
update</a>")
        # send email
        if len(email) > 5:
            msg = MIMEMultipart()
            msg['Subject'] = 'Welcome from Laundry Alert!'
            msg['From'] = '<YOUR EMAIL>'
            msg['To'] = email
            body = MIMEMultipart('alternative')
            html = 'Click here to return to the app... '<YOUR WEBSERVER>'=' + unitid +
'&action=check&sessionid=' + str(math.floor(sessid))
            body.attach(MIMEText(html,'html' ))
            msg.attach(body)
            s = smtplib.SMTP('localhost')
            s.send_message(msg)
```

```
            s.quit()
        if len(cellno) >= 10:
            voice = googlevoice.Voice()
            voice.login(email='<YOUR EMAIL>',passwd='<YOUR PASSWORD>')
            voice.send_sms(cellno, 'Welcome from Laundry Alert!')
# User clicks the link from the Email pointing to their session - shows last 5 values
read
if action == "check":
    print('Checking recent activity...<br>')
    stat = "DNE"
    crsr.execute("""select unitid, sessionactive, lastactivity from session where
session.sessionid = %s""",(sessionid))
    rows = crsr.fetchall()
    sunitid = 0
    for row in rows:
        if row is None:
            stat = "DNE"
        else:
            ct = 0
            sunitid = row[0]
            sessactive = row[1]
            lact = row[2]
            if sunitid is None:
                stat == "DNE"
            else:
                if sessactive == 0:
                    stat = "END"
                else:
                    stat = "ACTIVE"
    if stat == "DNE":
        print('<br><b><i>Session does not exist</b></i>')
    if stat == 'END':
        print('<br><b><i><font color=red>Your laundry completed at:<br>' + str(lact) +
'</font></b></i>')
    else:
        print ('<b><i><font color=red>Laundry still in process</font><br><a
href='<YOUR WEBSERVER>'+ unitid + '&action=check&sessionid=' + sessionid + '>Click to
Refresh</a>')
        print ('<table border=1><tr><td colspan = 3><b><i><font color=red>' + fnts +
'Last 5 values:</td></tr>')
        print ('<tr><td><b>' + fnts + 'Timestamp</td><td><b>' + fnts + 'Current
Draw</td><td><b>' + fnts + 'Current Leak</td></tr>')
        crsr.execute("""select timestamp, currentdraw, currentleak from unitdata where
unitid = %s order by timestamp desc limit 5""",(sunitid))
        rows = crsr.fetchall()
        for row in rows:
            if not(row is None):
                ts = row[0]
                cd = row[1]
                cl = row[2]
                currentd = (cd*(5/1023)*1000)/230
                currentl = (((cl*(5/1023)*1000)/230)*1000)
                currd = ('%.3f'%currentd)
                currl = ('%.3f'%currentl)
```

```
                print('<tr><td>' + fnts + str(ts) + '</td><td>' + fnts + currd +' A' +
'</td><td>' + fnts + currl +' mA' + '</td></tr>')
        print ('</table>')
# Default action is to initialize a new session for a device
if (action == "" or action == "Refresh") and not (unitid == "" or unitid is None) :
    # see if current active session for unit
    crsr.execute('select max(sessionid) from session where session.unitid = %s and
sessionactive = 1',(int(unitid)))
    for row in crsr.fetchone():
        if row is None:
            stat = "Open"
        else:
            stat = "Busy"
        # If refreshed, need to close the previous session
        print("<input type=hidden name=unit value=" + unitid + ">")
        if action == "Refresh":
            crsr.execute("""update session set sessionactive = 0 where unitid = %s and
sessionactive = 1""",(int(unitid)))
            stat = "Open"
        if stat == "Busy":
            print("<b><i>Unfortunately, that unit is in use at this time.  Try again
later.</b></i>")
            print('<br><br>If you are sure the previous user is done, click
here:<input type=submit name=action style="font-size : 75px; width: 100%; height:
100px" value=Refresh>')
        else:
            if action == "Refresh":
                print ('<font color=blue><i>Previous session was
closed</i></font><br>')
            print("<b>Unit " + unitid + " is available!<br><hr>")
            print('Enter your cell# for updates:<br>   <input type=text
style="font-size : 75px; width: 100%; height: 100px" name=cellno value="' + cellno +
'"><br> <br> <br>')
            print('Enter your email:<br>   <input type=text
style="font-size : 50px; width: 100%; height: 100px" name=email value="' + email +
'">')
            print('<br> <br>')
            print('<tr><td colspan=2 align=center><input type=submit name=action
style="font-size : 50px; width: 100%; height: 100px" value=Update></td></tr></table>')
else:
    if action == "":
        print("<b>Welcome to Laundry Alert!<br>Please scan a valid unit QR code to
begin")
print ("</body></html>")
cnxn.commit()
cnxn.close()
```