HOME ENERGY ADMINISTRATION TOOL (H.E.A.T.)

By

Edward Choi (echoi46)
Jae Min Song (json78)
JeeHaeng Yoo (jyoo46)

Final Report for ECE 445, Senior Design, Spring 2018
TA: Nicholas Ratajczyk

02 May 2018
Project No. 50

Abstract

Home Energy Administration Tool (HEAT) is a device that can be used within households to remotely monitor and control each appliance that is connected to it. It can be controlled from anywhere using our web application that shows its real-time energy consumption, daily usage comparisons and the estimated bill.

Contents

1. Introduction

Our proposed project was aimed to arouse consciousness to save energy within households. Our convenient web application showed the amount of real-time energy use on specific appliance so we expected people to become better knowledgeable of energy consumption, and hopefully save on their next bill. This chapter states the existing problem and how we aimed to resolve it.

1.1 Objective

According to the CIA World Factbook, the United States is the second largest energy consumer considering total use [1]. It is estimated that by 2088 we will run out of fossil fuels because energy reserves are clearly finite and the rate at which the world consumes them is increasing as the world's population increases. If we continue to use energy in this fashion, we will run out of energy eventually. We can stop pursuing fossil fuels by alternatively exploiting renewable energy sources, but this has to happen more quickly as energy resources dry up fast. While the world switches to renewable energy systems gradually, we can slow down the rate at which the energy is consumed within households. In order to slow down the energy consumption, it is first necessary that people become aware of how much energy they use daily, primarily because people tend to overuse energy unconsciously. However, it is often difficult to realize exactly how and how much energy they could save because they are uninformed of the specific amount of energy consumed by each appliance within their household.
We aimed to solve this problem by creating a comprehensive administrative tool that provides detailed and quantitative information about each appliance's energy consumption to residents within households. The power of each appliance can be remotely controlled through a responsive web application we developed. Most importantly, it displays detailed statistics that helps the user keep track of energy consumption and make decisions towards energy conservation. The information we provide includes each appliance's energy consumption, estimated bill, and daily usage comparisons.

1.2 Background

Currently, no ordinary household has a device or tool to measure the energy consumption of each appliance at home. We believed that our designed system could be very informative towards arousing awareness of energy conservation. Some of the other existing measuring tools could only measure a single component and were quite expensive. However, our design was able to measure two or more appliances at the same time and were set up embedded behind the virtual walls we made. We wish people to become more aware of how much energy each appliance takes up of the total energy consumption and realize the necessity to economize energy by using our tool in the future.

1.3 High-Level Overview (Block Diagram)

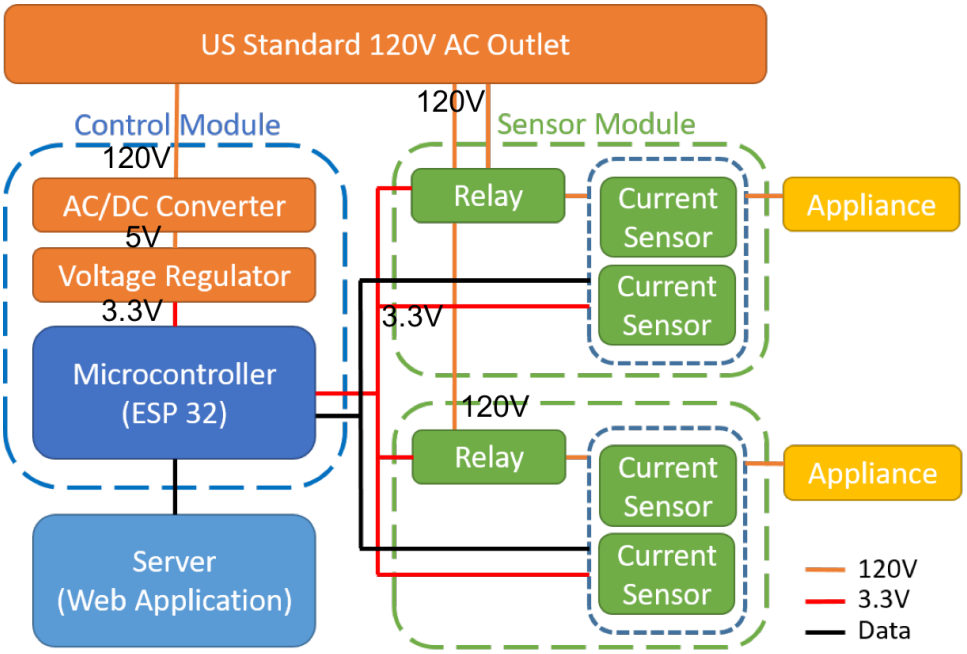

Figure 1 Block Diagram - Modular Design

Figure 1 shows the block diagram for the design of our system. We had two main modules, control module and sensor module, power supply and the server. The power supply was a US standard wall outlet which provided 120V AC and it supplied power to the whole system. The control module consisted of a AC/DC converter, voltage regulator and a microcontroller. A regular USB adapter was used for the AC/DC converter to convert 120V AC to 5V DC. Then, it was stepped down again into 3.3V DC so that it provided adequate amount of power to the microcontroller and the current sensors.

Each sensor module was composed of a relay and two current sensors which measured the current flowing through each appliance. With the relays, which worked as an electrical switch, we could connect or disconnect the current flowing through specific appliances by sending appropriate signals from the microcontroller.

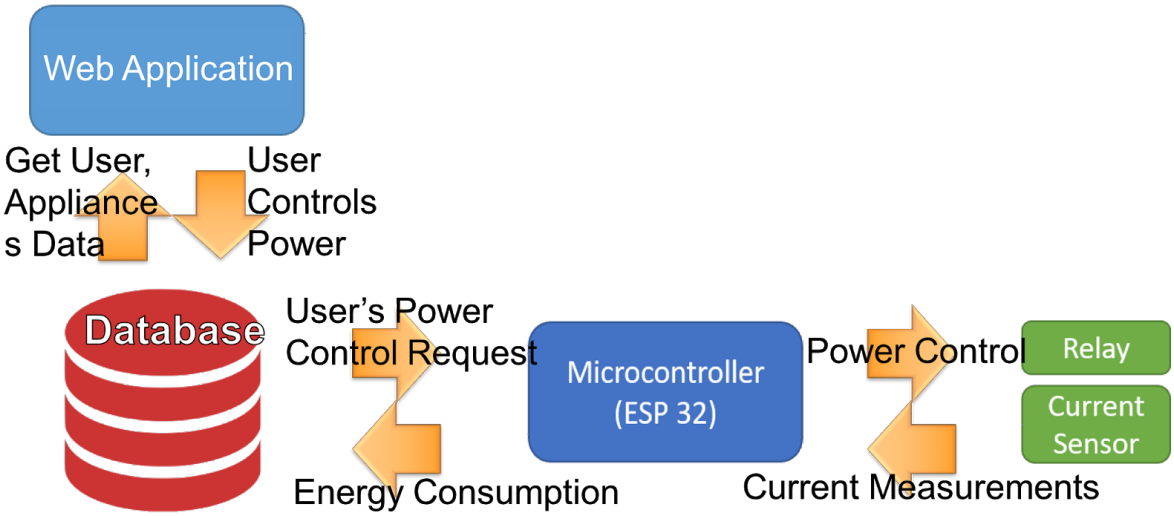

Figure 2 Block Diagram - Data Flow

Figure 2 depicts the data flow in our system. Database was the main part of the data flow, storing all relevant information regarding the connected appliances. Our web application received users' requests on whether to turn on or off a specific appliance. Once the web application modified the status on the database, the microcontroller obtained the updated status and sent signals to the relays to control the power of the connected appliances. When the power was on, the current sensor measured the current flowing through the appliances and sent the measurement to the microcontroller so that it could calculate the energy consumption and update the database accordingly. These data were reflected immediately on the web application to provide real-time statistics to users.

2 Design

This chapter discusses the detailed design decisions we made throughout the course of the project. It also describes each design module and components in detail and how we came up with our final design as opposed to the original one.

2.1 Physical Design

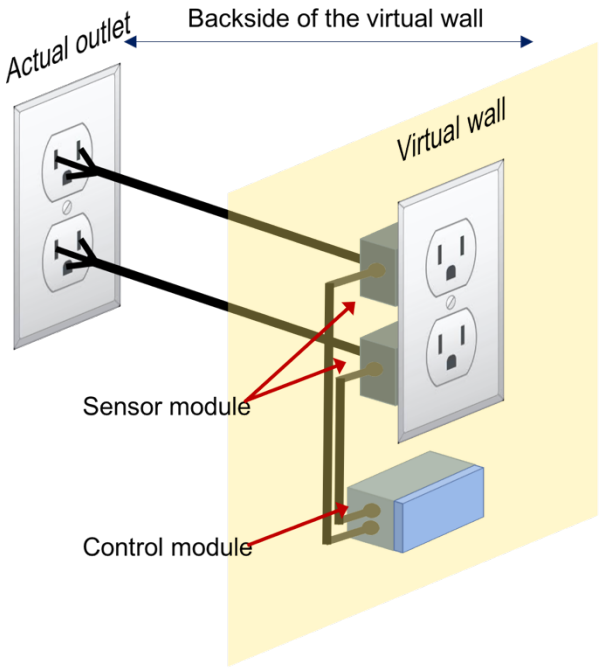

Figure 3 Physical Design Illustration

Figure 3 shows the physical design of our system. If we were to apply this to the real world, we want our system to be embedded behind the walls to improve applicability. In order to achieve this design, we made a fake wall and had to extend the power cords from the actual wall outlet to provide equal amount of power to our system. At first, we tried to embed everything on a single PCB board but since we were dealing with high voltage and high current, we had to separate the control module and the sensor module. Then, we needed to duplicate the sensor module to match the number of appliances connected to our device. The final design of our device including the web application is shown in *Appendix B Figure 8* and *Figure 9.*

2.2 Design Detail

As we developed our system, we encountered several technical challenges which caused our design to be modified.

We initially designed our system to have one sensor module that could account for more than one appliances. However, due to the safety concerns regarding high voltage and current flow on the same PCB boards, we decided to separate the sensor modules so that one PCB board could deal with a single appliance.

We also decided to put an additional current sensor on each sensor module to measure not only the current entering the appliance, but also the current that could possibly leak from the appliance. By measuring currents on two different ports, we could achieve better accuracy on current measurements and therefore provide more accurate energy consumption statistics.

Initially, we were planning to use a different microcontroller along with the ESP32 of which we ended up using solely. However, we found out that ESP32 could serve as a microcontroller as well as a WiFi connector so we simply removed the existing microcontroller from our design. Utilizing the ESP32 to its maximum capacity, we could simplify our design and yet accomplish the features we originally planned to offer.

One of the main features on our tool was the remote power control. At first, we had a difficulty figuring out the way to switch on/off the 120V power without applying an actual physical contact. Then, we came up with the idea of using relays to achieve such functionality. Relay worked as an electrical switch that allowed us to control 120V using small amount of supply voltage. Yet, another problem followed as the microcontroller could not output enough voltage to operate the relay. In order to operate the relay with sufficient amount of current, we implemented a subcircuit using bipolar junction transistor to amplify the voltage. After all, using the relays and BJT, we could successfully control the power of the appliance remotely.

2.3 Block Design Overview

Following descriptions of parts explains in detail how each part is used in our system. All the necessary circuit schematics as well as PCB board designs can be found in *Appendix B*.

2.3.1 Power Supply

The power supply which was used to provide power to our system was the typical 120V AC wall outlet. As shown in *Figure 3*, our device was connected to this outlet and let our own outlet provide an equal amount of voltage to each appliance.

2.3.2 AC/DC Converter

The AC/DC converter we used was the regular USB adapter that could step down and convert 120V AC to a sustainable 5V DC output. We chose this adapter so that we could use a micro USB cable to program our microcontroller using the Arduino IDE.

2.3.3 Voltage Regulator

The voltage regulator was used to step down further to 3.3V DC from 5V DC. We used the regulator that could output a minimum current of 600mA and a voltage in range 3.27V ~ 3.33V, which was sufficient to operate our microcontroller [3].

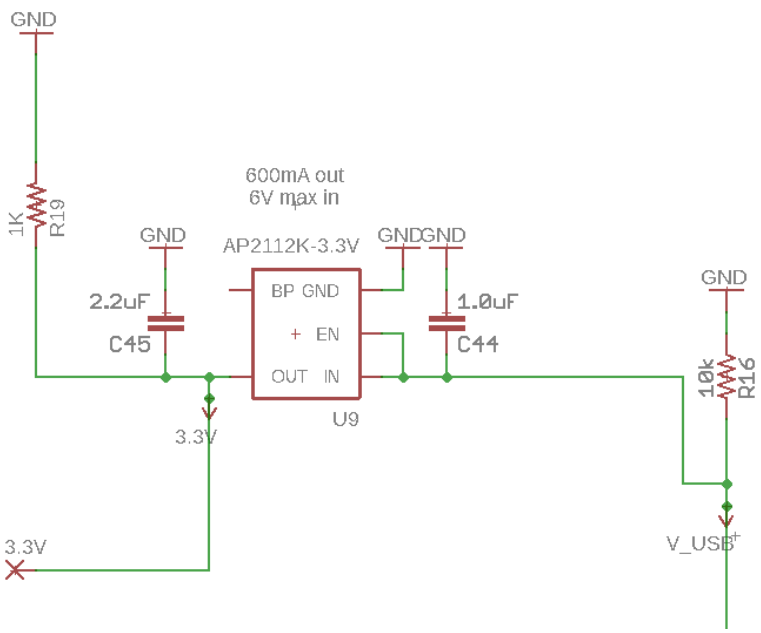

Figure 4 Control Module Schematic - Voltage Regulator

2.3.4 Microcontroller (ESP32)

The main decision factor on why we chose ESP32 as our microcontroller was its capability of WiFi communication. [4] We needed to setup the WiFi connection because the microcontroller had to send and receive data to and from the database through the http requests on the APIs we created. Along with the microcontroller, peripheral components such as the USB to serial converter, flash memory, antennas all assisted in analog and digital data communication. The ESP32 controlled the whole system by sending out appropriate signals to the web application's backend and the relay, and by receiving values from the current sensor and the database.

Our microcontroller received current measurements every ~17ms from the current sensor as its frequency was 60Hz. Then, it summed up the current measurements for 10 minutes, which we set up as an update period on our web application.

$$Energy[Wh] = \int_0^T VI(t) = 120[V] * \frac{1}{6}[hr] * \int_0^{1/6} I(t) \quad (1)$$

The energy consumption was calculated by using *Equation (1)* and was updated on the database, which then automatically was updated on the web as well.

2.3.5 Bipolar Junction Transistor (BJT)

We made a bipolar junction transistor circuit by connecting the base to the GPIO of the microcontroller, the emitter to a 5V voltage supplier, and the collector to the ground. When the 3.3V supply voltage was given, we were able to obtain the output voltage less than 2V and when 0V was given, we could obtain voltage close to 5V, which was enough to operate the relay.

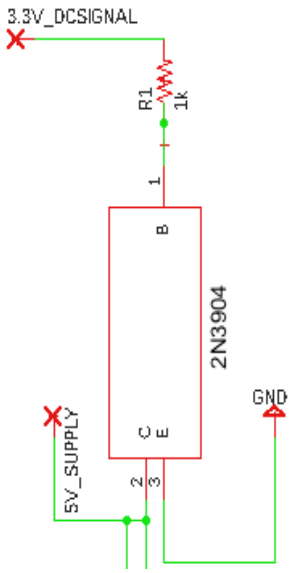

Figure 5 Sensor Module Schematic - BJT

2.3.6 Relay

In order to let user remotely control the power of each connected appliance, we used the relays that could operate as an electrical switch that connected/disconnected the 120V AC current flowing through the appliances. [5] Since voltage greater than 3.75V was required to operate the relay while our microcontroller only outputted 3.3V, we had to put a BJT in between the microcontroller and the relay to amplify the voltage to 5V.

2.3.7 Current Sensor

The current sensor that we chose could measure up to 20A, and we assumed that no more than 20A of current could be drawn by an appliance. In our system, they were powered by the microcontroller to measure the amount of current drawn by an appliance. We used two current sensors per appliance to measure the current on live and earth socket on electric outlet in order to account for the current leakage to ground. The measurement was then sent to the microcontroller and was stored in the database. The data received on the microcontroller from the current sensor had a value proportional to the ratio of the voltage output to the voltage supply as in *Equation (2)*. Then, according to the datasheet, the current can be calculated by using *Equation (3)*. [6]

$$Vout = \frac{(analogRead - 2048)}{4096} * V_{cc}, where\ V_{cc} = 3.3V \tag{2}$$

$$I = Vout * 0.5 * \frac{1A}{66mV}(sensitivity) = \frac{(analogRead - 2048)}{4096} * \frac{0.5 * 1A}{66mV} \tag{3}$$

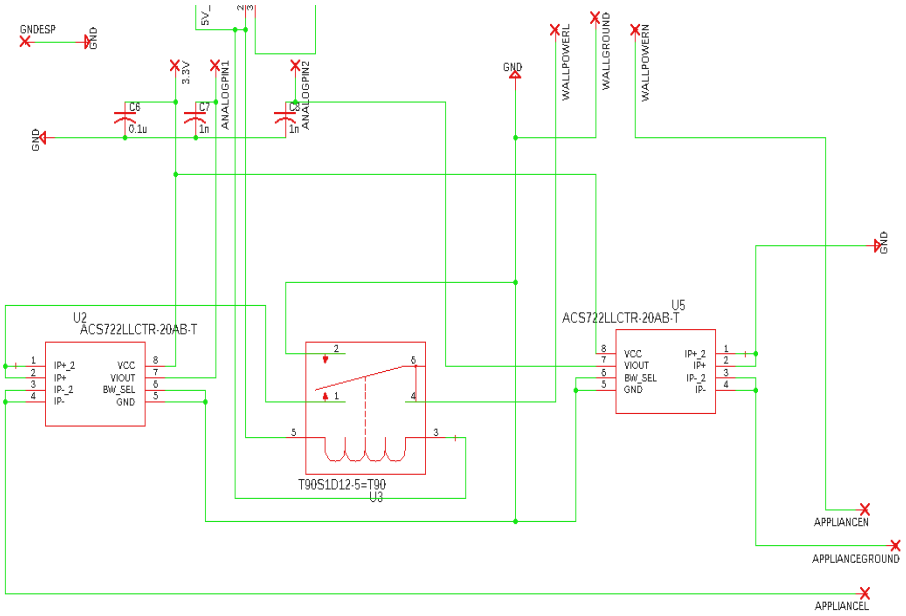

Figure 6 Sensor Module Schematic - Relay and Current Sensor

2.3.8 Database and Web Application

Through the backend APIs that enabled connection to our database, our web application transmitted and received data to and from the microcontroller. We used MongoDB to store and query the users' data and built the APIs on Express which is the Node.js framework to connect to the server. For each appliance connected to the system, the database stored unique id, name, power status, remote control status, and daily energy consumption. We used id and name to identify the appliance within the database. Power and remote control status indicated whether the appliance was on or off and whether it should be turned on or off, respectively.

When the energy consumption was queried on the web, we could also calculate the estimated cost in the frontend using the *Equation (4)*. We calculated the average hourly rate of cost in *Wh* from the *Ameren Illinois* bill because the actual rate differs throughout the period of time of the day.

$$Estimated\ Cost = Energy[Wh] * Averaged\ rate(\$0.00011467) \quad (4)$$

3. Design Verification

The Requirements and verification table for our system can be found on *Appendix A.*

3.1 Power Supply

We required the wall outlet which is the power supply to output the same 120V AC with a RMS voltage of range 114V AC to 126V AC [2]. When we checked with a voltmeter we observed a stable 119.0V AC measurement.

3.2 AC/DC Converter

We had a USB adapter which converted 120V AC to 5V DC. We wanted a stable output measurement of 5 ± 0.2V DC. When we connected each end of multimeter to the 5V output and the ground, we measured 4.83V which met our requirement range.

3.3 Voltage Regulator

The voltage regulator we used needed to output a DC voltage in range from 3.27V to 3.33V resulting from a 5V DC input. We verified this with our multimeter, reading 3.279V, which was acceptable.

The voltage regulator also had to output a minimum of 500mA to provide enough power to our microcontroller. According to the voltage regulator datasheet, it should output a minimum of 600mA, which is sufficient to operate our microcontroller [3]. We checked this data by observing our microcontroller's functionality in the end, which verified that the current supplied by the voltage regulator was enough.

3.4 Microcontroller

We built our own ESP32 board with the microcontroller chip soldered along with other parts such as the USB to serial converter, flash memory, voltage regulator, WiFi antenna, and etc. We had the USB to serial converter to allow our microcontroller to be programmed with an Arduino code. We wrote the to print "Hello World" onto our screen to check this functionality. The program printed out "Hello World", proving that our ESP chip could be programmed properly.

Also, we built a resistive circuit with a LED and connected a relay to the circuit. We had our microcontroller output 'high' and 'low' signals from the general input/output pins. As the 'high' and 'low' signals were sent to the relay, the LED would light on and off, which showed that our microcontroller had general input/output pins working. Also, we measured the voltage of the output pins with the multimeter and got 3.203V when the signal was high and 0V when low, which met our requirements.

The ESP32 was able to connect to the WiFi. We had an antenna connected to the system to allow us to use the WiFi functionality. We turned on our hotspot from our smartphones and made a successful WiFi connection through the ESP32.

We built a simple resistive circuit and supplied it with 3.3V output from a power supply. We had our microcontroller connected to the ends of the resistors, which read analog signals of the voltages. The microcontroller read readings based on the ratio of its bits. Converting the analog signals, the microcontroller showed that the voltage readings were 3.27V, which met our requirements.

3.5 Bipolar Junction Transistor (BJT)

We built a BJT circuit connecting the base with an output pin from the microcontroller and a 1K ohm resistor. We had a 5V supply on the collector of the BJT and ground on the emitter. We generated 'high' and 'low' signal outputs from the microcontroller and measured the collector voltage with a multimeter. We measured 1.8V when the signal from the microcontroller was 'high' and 4.9V when it was 'low,' which met our requirements.

3.6 Relay

As for the relay, we connected the coil of our relay to the collector of the BJT from our previous circuit testing for the BJT. We had a resistor and an LED connected to the current flow of the relay. As we sent 'high' and 'low' signals from the microcontroller, the relays switched on and off, turning the LED on and off. As we measured the coil voltage, the LED lit up when the coil voltage was 1.8V and turned off when the coil voltage was 4.9V. Our relay was a normally closed relay and from the tests that we had made, it met our requirements for the system.

3.7 Current Sensor

One of the most important requirements of our system was current measurement. We required the current sensors to measure the current with an error of less than 3%. However, when we obtained the measurements of the connected laptop charger on the microcontroller, our sensors did not output correct values. Instead, it kept sending largely fluctuating values that were hard to interpret. To verify the performance of sensors, we tried to scrape the serial output from the sensor and plot it using a python script. When we plotted the measurement, we observed the measurements showing sinusoidal waves but fluctuating too much. We applied a Savitzky-Golay smoothing filter to the signal to see if we could rely on the measurements, and we were able to obtain clear sinusoidal waves. We concluded that our current sensors were working correctly, but the measurements were affected by a strong

electromagnetic field interference that was generated from the large current flowing by the sensors on the same PCB board.

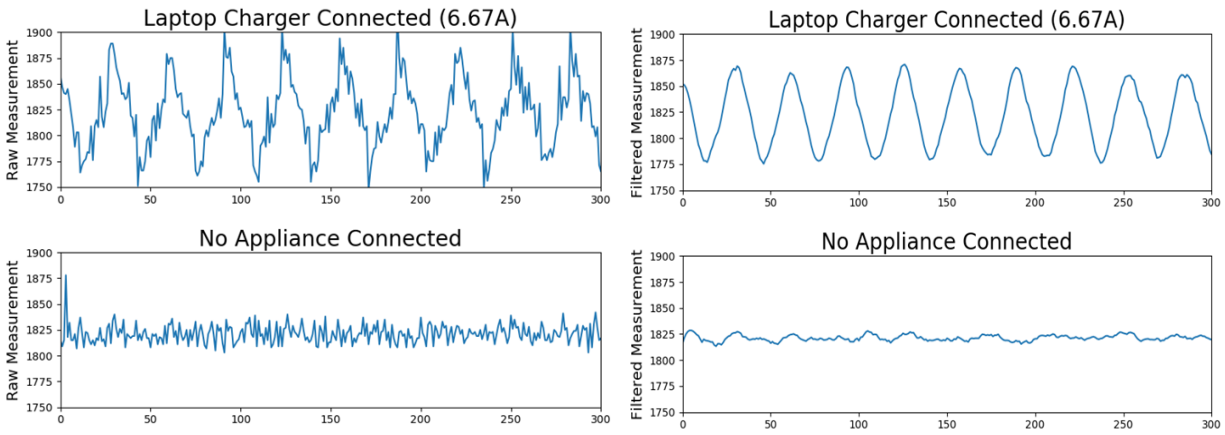

Figure 7 Comparison on Raw Current Measurement and Filtered Current Measurement

Applying the filter definitely removed the fluctuation on the measurement, however, we could not apply the filter on our microcontroller. Instead, to obtain reasonable measurements, we came up with two steps of calculations for the current measurement. We assumed that the interference resulted the measurement to be slightly raised, so we reset the offset of current calculation. Next, we ignored every measurement below the offset and used the positive measurements only. Equation 1 shows the formula for the current measurement calculation we used in the system.

$$I(t) = \frac{Analog\ Measurement}{Max\ Analog\ Output} * V_{cc}[V] * \frac{1}{period} * \frac{1}{0.066}[A/V] \quad (5)$$

First two terms in Equation 1 scales the analog measurement. To calculate the current per period, we averaged the measurement by dividing the measurement by the length of one period. The last term indicates the sensitivity, which indicates how much voltage output from the sensor is measured to be 1A. We used the sensitivity given on the datasheet of the current sensor. When we calculated the current using the equation, the measurement on the laptop charger, which was supposed to be measured to be 6.67A, showed 6.61A. When nothing was connected, the calculation gave 0A. Although we failed to measure an exact current, we successfully identified the measurement within a reasonable range.

4. Costs and Schedule

4.1 Costs

4.1.1 Cost by Parts

Table 1 Costs by Parts

Part	Part Number	Cost	Quantity	Total Part Cost
Microcontroller	ESP32-d0wdq6	$5.88	1	$5.88
Voltage Regulator	AP2112k-3.3	$0.55	1	$0.55
Power Relay	T90S1D12-5	$5.54	3	$16.62
Current Sensor	ACS722LLCTR-20AB-T	$4.886	5	$24.43
Bipolar Junction Transistor	2N3904	$0.77	2	$1.54
GFCI Outlet	-	$9.99	2	$19.98
USB Cable	-	$2.99	1	$2.99
Power Cords	-	$3.99	2	$7.99
Crystal	26MHZ	$0.34	2	$0.68
Crystal	32.768kHZ	$0.56	2	$1.12
IC FLASH	32MBIT 133MHZ	$0.97	1	$0.97
Switch	-	$0.74	2	$1.48
Antenna	-	$2.59	1	$2.59
Micro USB connector	-	$0.42	2	$0.84
IC USB Serial full UART	-	$2.12	1	$2.12
Semiconductor	MBT3904	$0.22	2	$0.44
Control Module PCB	-	$23	1	$23
Foam boards	-	-	2	$16.21
Total Cost	$140.63			

4.1.2 Labor

$$labor = 3[ppl] * 40[\$/hr] * 20[hr/weeks] * 15[weeks] * 2.5 = \$90{,}000 \quad (6)$$

$$Grand\ Total = labor + Parts = \$90{,}140.63 \quad (7)$$

4.2 Schedule

Table 2 Schedule

WEEK	Jee Haeng	Jae Min	Edward

02.19.2018	Definitize the system design and finalize the design document.	Visualize our project and finalize the design document.	Finish first design of the circuit and prepare for mock design
02.26.2018	Review and finalize schematics and pin connections.	Prepare for design review and modify our document.	Prepare for design review and finish up design
03.05.2018	Prepare for presentation and start studying web server.	Work on circuit layouts, configure microcontroller connections	Order all parts and start making current sensor PCB board
03.12.2018	Review and finalize the PCB design.	Finalize PCB design and order PCB	Test parts that have arrived and start making ESP board.
03.19.2018	Write code to program the microcontroller (control I/O).	Start programming for microcontroller	Continue working on current sensors and set up relays
03.26.2018	Verify if all components work properly.	Modify PCB and order more if we need to	Finalize PCB and order PCB
04.02.2018	Write code to program the microcontroller.	Start to develop web application	Finish soldering PCB and testing.
04.09.2018	Configure all computations on microcontroller.	Work on data communication modules, create backend APIs	Finish testing PCB and prepare for mock demo, presentation
04.16.2018	Debugging.	Finish up frontend and backend	Finish all requirements and verifications for demonstration
04.23.2018	Prepare for the Final Presentation	Prepare for final presentation	Prepare presentation and final paper.

5. Conclusion

5.1 Accomplishments

Overall, most parts of the system worked as we designed. Our current measurement, although it fell within a reasonable range, was not accurate enough. The rest of the system- web application, database, backend API, microcontroller and relays- achieved its desired functionalities and we were able to accomplish the overall goal of the system.

5.2 Uncertainties

Uncertainty remains on the project regarding the calculation of the correct current measurement. We came up with different method of calculation to fit the measurement within the reasonable range, but the calculation cannot be theoretically justified. Also, we could not identify the quantitative analysis of electromagnetic field interference, which could be a clue to build a formula that can result in higher accuracy in calculation of current measurement.

5.3 Ethical considerations

We were responsible for the information that was sent to the public through our product. We pledged to the IEEE code of ethics [7] and in order to follow the code #1, we tried to implement the sensor module separately on each outlet because if we merged more than three sensor circuits in one circuit, it would have drawn more than 60A of current. If we continued to work on the circuit that can go anywhere between 60A to 100A, it would create severe voltage fluctuations which could endanger potential customers. Our design reduced the risk of current overloads but that did not imply total freedom from dangers. Since our design used high voltage of 120V AC which could cause electric shock and fire hazards, we must always be careful when dealing with power cables and wires and check if the power is turned off before we touch the circuit. One way we prevented the hazard was to use an appropriate gauge of wire that could handle high current. It was also important to turn off the power whenever the system was not in use, even for a short period of time.

As the code #3 states, we have been honest and realistic with our measurement data. Consumers will mostly utilize the tool to monitor their total electricity bill and expect it to be close to the actual bill. Therefore, we could not just arbitrarily estimate the cost or fake data. Especially, since we wanted to supply this product with the main objective being “energy conservation,” we did not want to supply this product to people if our system required high energy. Therefore, we aimed to create our main control unit with less than 4.2 W of total power consumption with additional 1.2W per each connected appliance. As a long-term goal, we expect our system to “improve the understanding by individuals and society” (code #5) by our informative and intuitive web application providing customized data analysis for individual users.

To ensure that we followed the code #6, “to undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations,” we were required to complete our laboratory safety training, PCB workshop, and soldering assignment as part of our project. As for the

full disclosure of pertinent limitations, we would state that using our device may cause safety hazards when not used properly or when the instructions we provided are not followed.

5.4 Future work

Our current sensor functionality was not ideal as we had expected. As for future work, we our thinking of having an alternative current sensor that is less affected by electromagnetic interference and that would fluctuate less. Another way that we can solve our problem is to have our current sensor be far away from the 120V AC outlet to reduce the fluctuation.

As we plan to have all the appliances in a household to be measured, we will need to increase our ESP32 flash memory size to be able to hold all the appliance's data and to control them. Also, the ESP32 will need to have a stable connection of WiFi in order to control all the appliance in a household.

References

[1] cia.gov, "*The WORLD FACTBOOK*", 2015. [Online]. Available: https://www.cia.gov/library/publications/the-world-factbook/rankorder/2233rank.html#us [Accessed: 04-FEB-2017]

[2] "*American National Standard for Electric Power Systems and Equipment—Voltage Ratings (60 Hz)*," *ANSI*, vol. C84.1, no. 2016, Oct. 2016.

[3] "600mA CMOS LDO REGULATOR WITH ENABLE," *Diodes*, Jun-2017. [Online]. Available: https://www.diodes.com/assets/Datasheets/AP2112.pdf. [Accessed: 12-Apr-2018].

[4] espressif.com, "*ESP 32 Data Sheet*", [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf [Accessed: 07-FEB-2017]

[5] Potter & Brumfield, "*T90 Series, 30A PCB Relay*," mouser, Feb-2014. [Online]. Available: https://www.mouser.com/ds/2/418/NG_DS_1308242_T90_0214_T90-718891.pdf. [Accessed: Feb-19AD].

[6] High Accuracy, Galvanically Isolated Current Sensor IC With Small Footprint SOIC8 Package. (2017). Allegro Microsystems, p.4, 6.

[7] Ieee.org, "*IEEE IEEE Code of Ethics*", 2016. [Online]. Available: http://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: 05-FEB- 2017]

Appendix A Requirement and Verification Table

Table 3 System Requirements and Verifications

Power Supply [3 Points]	
Requirements	**Verification**
1. The power supply should provide 120V AC with a RMS voltage in range 114V AC ~ 126V AC . [3]	1. Use a voltmeter to measure the voltage from the wall outlet.
AC/DC Converter [3 Points]	
Requirements	**Verification**
1. The converter must output a voltage value that is within 50.2V DC. [3]	1. a. Connect output to oscilloscope. b. Apply voltage. c. Check if the voltage has less than 4% of error.
Voltage Regulator [6 Points]	
Requirements	**Verification**
1. The regulator should output a voltage range from 3.27V to 3.33V [3]. 2. The regulator should output current of at least 500mA to supply the microcontroller. [3]	1. a. Connect output to multimeter. b. Apply voltage. c. Check if the voltage is within3.30.03V. 2. a. Check if ESP32 works.
Microcontroller [16 Points]	
Requirements	**Verification**
1. It should be able to be programmed with the uploaded code via Arduino IDE. [4] 2. The GPIOs can input/output high/low signals. [4] 3. The microcontroller can connect to Wifi. [4]	1. a. It is able to print “Hello World” onto the serial monitor. 2. a. Build a resistive circuit with a LED. b. Output high/low signal with the delay on Arduino IDE. c. Check whether the LED lights

4. The microcontroller should be able to read analog signals and convert into digital data. [4]	on/off. 3. a. Upload the code that sets up a Wifi connection. b. Turn on the Hotspot on the smartphone. c. Check if the device is connected on the smartphone. 4. a. Build a simple resistive circuit. b. Supply voltage between 0-3.3V. c. Read both ends of the resistor as analogRead. d. Calculate the voltage between the resistor.
Relay [8 Points]	
Requirements	**Verification**
1. The relay switch should be able to control the current flow depending on the voltage. It should open when the supply voltage is higher than 3.75V and close when less than 2V. [8]	1. a. Build a resistive LED circuit attached with the relay. b. Supply the relay with voltage higher than 3.75V. c. Check if the LED lights off. d. Do the same thing with voltage lower than 2V and check if the LED lights on.
Current Sensor [10 Points]	
Requirements	**Verification**
1. The current sensor should be able to identify current with less than *3%*of error. [10]	1. a. Build a simple resistive circuit. b. Use resistors to adjust input current to be 20A. c. Connect microcontroller to computer and print input from the sensor to verify that measurement matches the actual current with less than 3%.
BJT [4 Points]	
Requirements	**Verification**

1. BJT should be able to amplify the low signal voltage(0V) to the voltage that is higher than 3.75V. [2] 2. BJT should be able to reduce the high signal voltage(3.3V) to the voltage lower than 2V. [2]	1. a. Make a BJT circuit by connecting the base side with the GPIO from the microcontroller and 1kohm resistor. b. Connect collector side with a voltage supplier(5V). c. ground the emitter side. d. Check if high(3.3V) supply voltage will output voltage less than 2V when measured from the emitter side and the ground. 2. a. Do the same thing as in #1. b. Check if low(0V) suppply voltage will output voltage close to 5V.

Appendix B

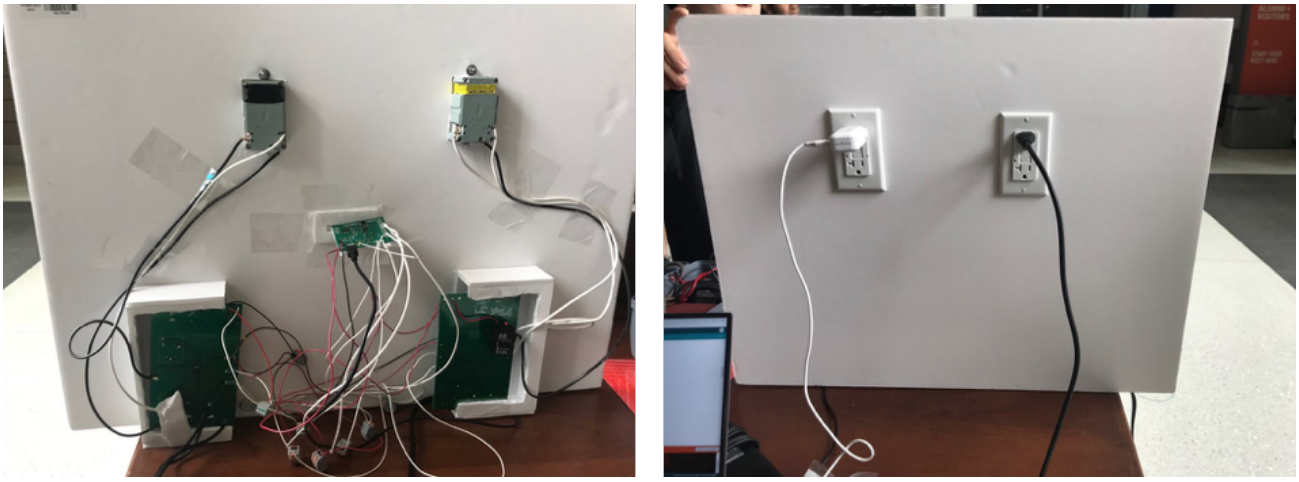

Figure 8 Final Design - Whole System

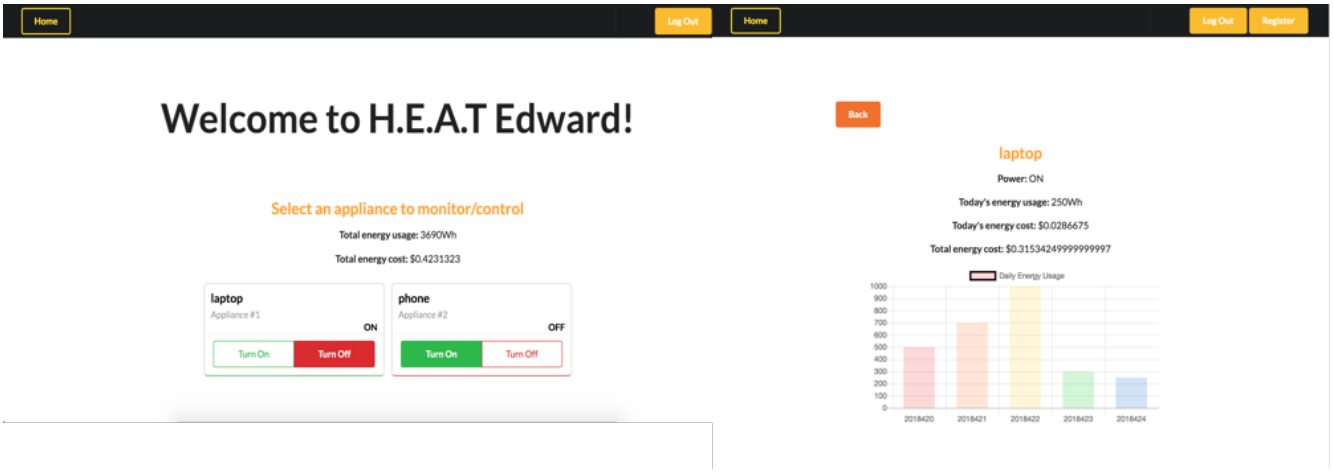

Figure 9 Final Design - Web Application

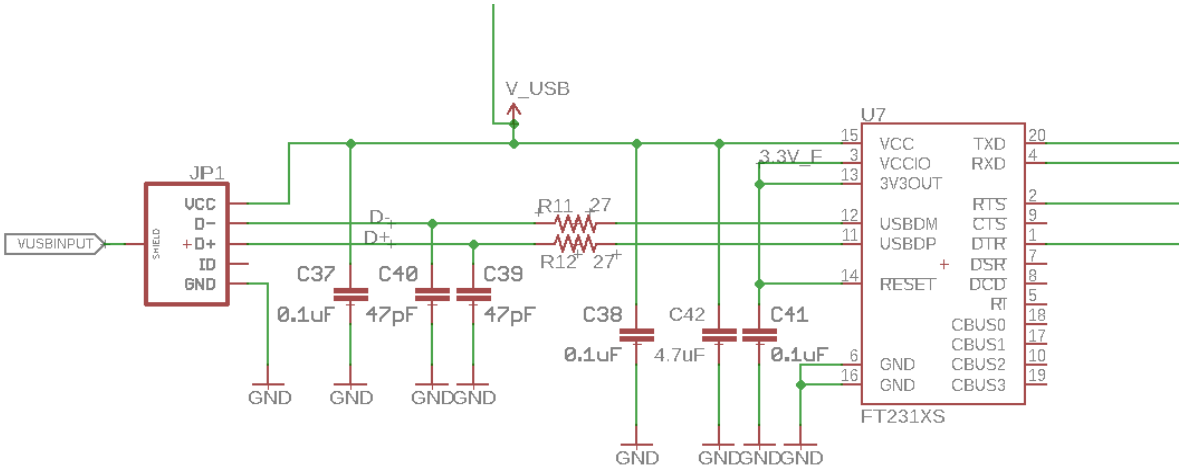

Figure 10 Control Module Schematics - USB to Serial Converter

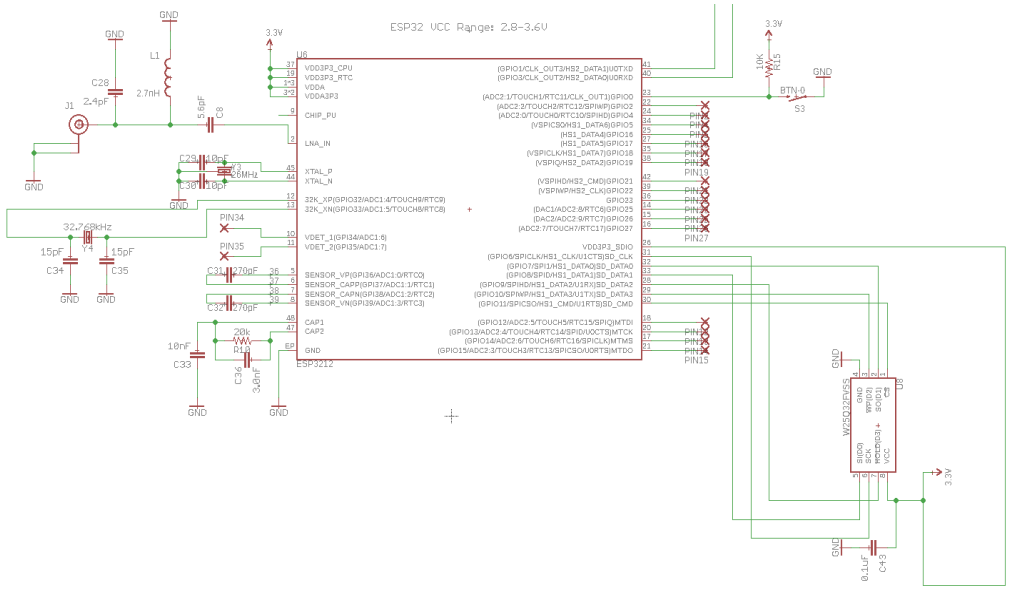

Figure 11 Control Module Schematics - ESP32 with spi Flash and Antenna

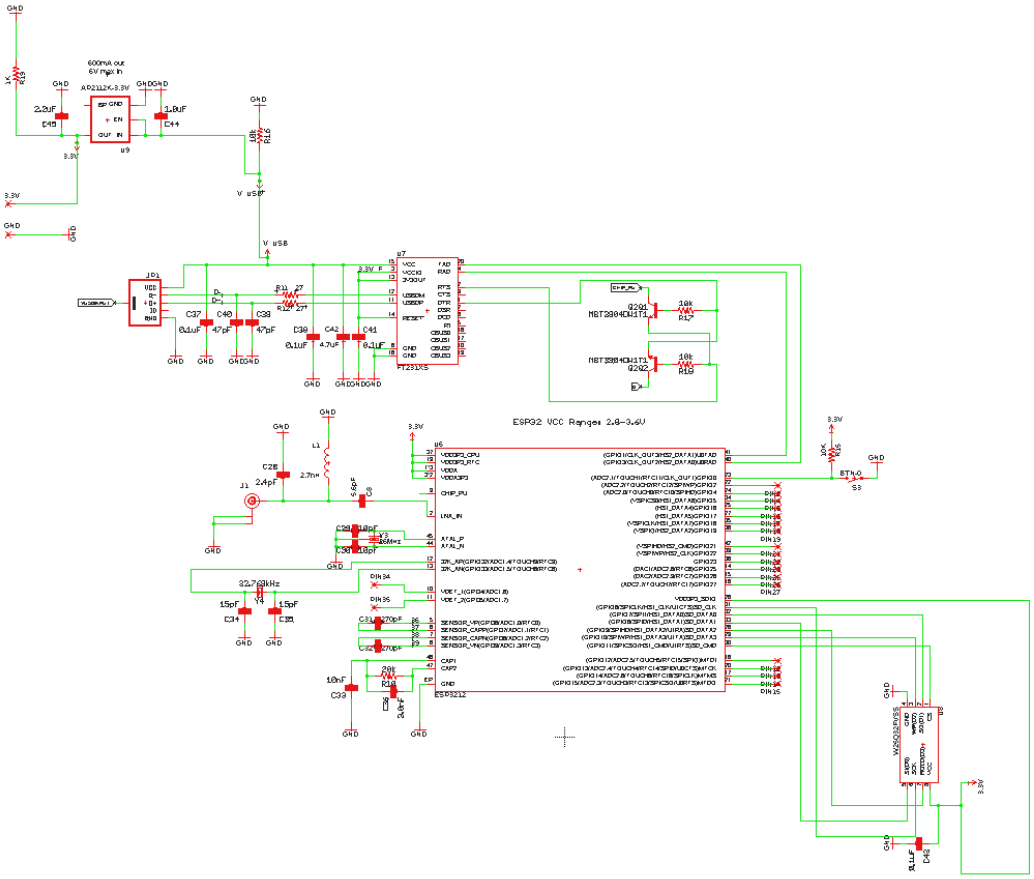

Figure 12 Control Module Schematics

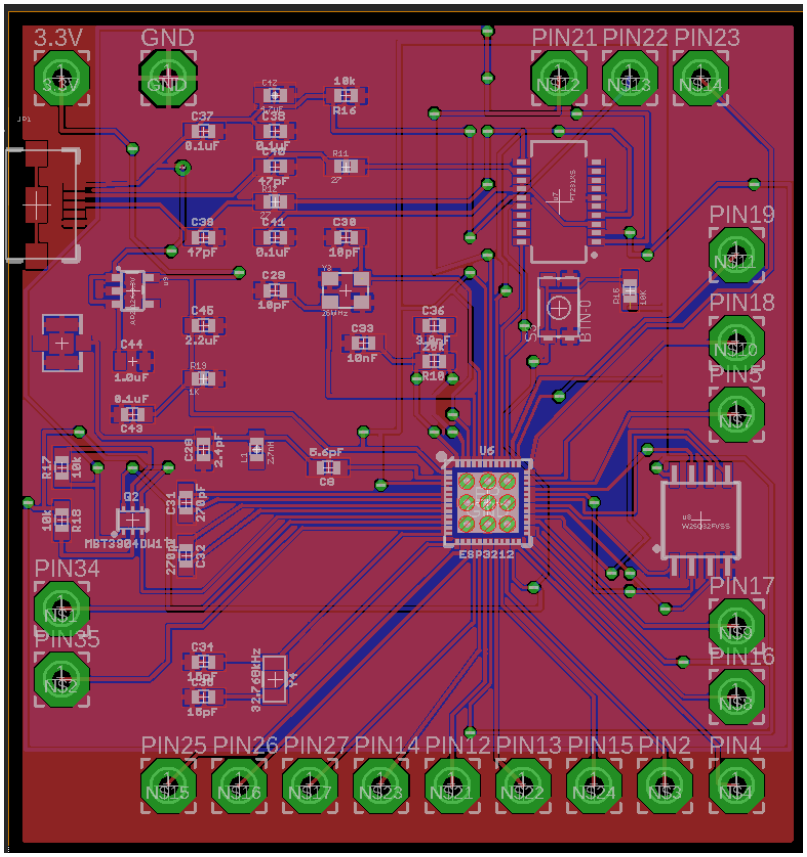

Figure 13 Control Module PCB Board

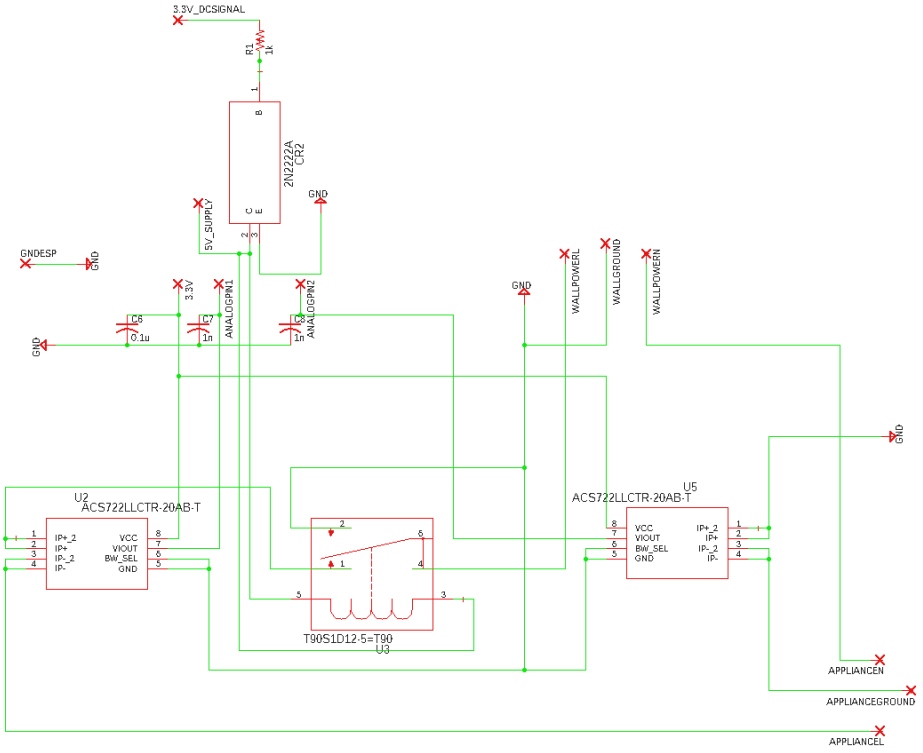

Figure 14 Sensor Module Schematics

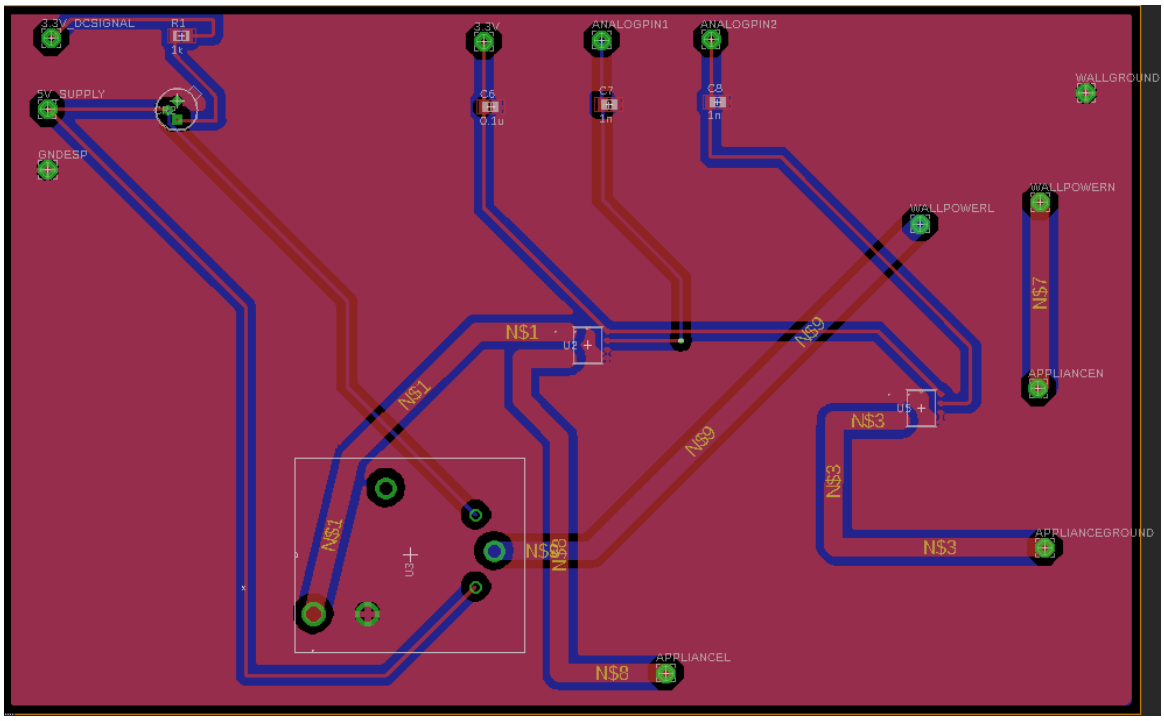

Figure 15 Sensor Module PCB Board