Automatic Pet Food Dispenser
Group 25

Members:
Nick Nabors, CpE
Bao Nguyen, CpE
Mehrob Farhangmehr, CpE
Hamid Igliou, EE

Senior Design 2 - EEL 4915 - Summer 2021
Due: August 03, 2021

Table of Contents

List of Figures

List of Tables

1. Project Summary

For as long as we can remember people have always had pets to help them through many challenges in life. As of current times pets are more as a luxury and a friend that we have at home. Having a pet has many challenges when it comes to keeping them healthy. Some issues we face when it comes to health are physical health, making sure they get enough exercise, mental health, making sure we give them enough attention and support to feel comfortable. Another big issue when it comes to pets is feeding them the right foods and enough food.

The problem we are really focusing on here is the food aspect of owning a pet. A lot of companies now are making a lot of healthy foods for pet owners to buy. The main focus of our project is being able to get nutritious food to our pets. We are all familiar with feeding our pets or, at least, have a friend that does. We have noticed that, sometimes, pet owners are out of their house and need to go home whenever they have to, or forgot to, feed their pet. This can be an inconvenience at times when we do not want to leave where we currently are (ex: date, out with friends, etc).

Our solution to this problem is an automatic food dispenser, with many helpful features, that makes life, for the average pet owner, easier. With everyone now working and having the ability to travel and go out more with cars, this would be the perfect solution for many pet owners. With an automatic food dispenser for their pet they can do whatever is needed and not have to worry about their pet not ever having food. An issue that could arise from leaving a bunch of food out for our pets is overeating and obesity, to fix that we would like to implement a portioning feature, that would allow the owner to select a portioning size they think would be best for their pet.

Roughly sixty-seven percent of United States households are pet owners, which is a really big audience already not including many other parts of the world. With many of the developed countries where both male and female individuals are independent and working there is a great need for a solution to help feed their pets. The reason our solution is a great way to solve this situation is because there are many ways this can be implemented and improved upon in the future.

Currently there are projects also similar to the one we have now presented but with how technology is rapidly increasing there are many different ways to tackle the problem and make the solution better and better, each being unique from each other.

In conclusion there are a lot of pet owners in the world that face many of the same struggles, one being finding a way to feed their pets while balancing their busy lives. The way we plan on doing that is creating an automatic pet feeder that has many unique features that would help users customize the feeder to their pet. There are some designs out in the world already but we plan on attacking this problem in our unique way, and having features that we thought were important and are missing from other solutions that others have come up with for this problem.

2. Project Description

The goal of this project is to provide the end user with automatic/manual control of when, and how much, food is dispensed for their pet. This can be achieved through various methods and features that are built into the design. The design will be easy to use and can be used remotely, or physically, at the dispenser through buttons and/or a menu.

Although there might be few similar designs out there, our design will be unique in many ways. Our design will provide the pet owner with peace of mind wherever they may be. With a few clicks from a mobile phone, they should be able to feed their pet and not worry about rushing to get home from work or a trip in order to feed their pet. Our design will also allow the pet owner to control the amount of food dispensed to the pet. The owner will be able to provide a specific portion of food to the pet; the owner can decide if the pet should get a certain amount of portions for a meal or just one portion for a snack.

Furthermore, our design comes with a schedule that can be programmed by the owner to dispense at certain times. Through this option, the owner can decide when their pet should be fed while they can also control the pet's diet.

The owner will be able to remotely control the automatic pet feeder whether it be through bluetooth, wifi, and/or zigbee. The main goal of our project is to create a device that will help any pet owner feel more secure leaving their homes without the fright of forgetting to feed their pets, and feeling the need to always be coming home just to put out food. We want to help all pet owners have the ability to live their lives as well as taking care of someone close to them at home.

The reason we chose this project is because we all have or at least know the struggle of having a busy schedule while also owning a pet and the amount of responsibility that comes along with that. Even though there are a few other designs out there on this topic we want to make ours a little different in a way we think would be better than the current models such as range, power consumption, physical appearance, and much more. Having a pet and feeling the struggle of this problem also pushes us to develop something robust and useful not only for us to make sure our pets are also okay being home for a long period of time.

Some minor, but unique and important features we would like to add to our design that seem to be lacking or missing from other designs that we have seen are including an LED that allowed the user to know if the food was running low or not inside of the tank. Another feature that is similar to this for the mobile application there can also be notifications to let the user know when food is running low from a remote place, for example, if they were at work.

There are many things that we have planned for our project and many things we want to add, but these here are the main ideas we plan on implementing for now. There may be changes and additions as we progress through the design and creation of the product.

2.1 Project Requirements

In order to build such a project we will need multiple hardware parts that will be controlled by a software program. The first requirement for this project would be the availability of internet and a mobile phone. Both will establish a connection between the pet feeder and the owner's mobile device through the web. We will also design a software interface through which the owner can control the feeder. We will also need to establish a wireless connection between the feeder and the owner's router such as a bluetooth that will enable wireless communication between the feeder and the mobile device. These devices will be connected to a programmable controller that will receive commands from the owner's mobile device and send signals to the hardware in order to execute these commands. This controller will most likely be embedded in the pet feeder and connected to a PCB that will control the hardware.

When it comes to hardware, we will design a PCB that connects all the pet feeder hardware elements. This will include a power supply that will power all the elements of the feeder; this will be accomplished with a step down transformer. Once the owner plugs the feeder into an outlet, the step down transformer will convert the supplied power into a lower voltage that will provide adequate power to all the components inside the feeder. We are unable to provide power specifications at this time such as the voltage rating or whether we are using AC or DC voltage since we are at an early stage of the project. These specifications will be determined later on when other critical parts of the feeder are specified.

Another hardware component that would be very critical to the functionality of the feeder is a mechanical motor. The motor will receive a voltage signal from the PCB to open and close a gate through which the food portions will be dispensed. The PCB will in turn be controlled by the programmable controller. The motor should be able to rotate back and forth in order to accomplish its purpose, which is to swing the gate back and forth. It should also have a quick response capability in order to dispense the right amount of food and not exceed what the amount that the owner specified.

We will also need the automatic pet feeder to be able to notify owners when the food is running low and make sure the portioning of the means are close to the expected servings. To be able to do these thing we will need to have a sensor of some kind to be able to measure food levels in the container and once it has met a certain level that we would consider low, find a way to notify the owner that the food is running out whether that be an LED on the physical device or a notification on their mobile device. The device will also be able to change LED to color to indicate whether the food tank is full or running low on food.

For portioning we will need a sensor for weight and have measured portion sizes for what animal would be using the device. Once that is determined we would need the sensor to communicate with the MCU to close or open the gate controlled by the motor that was mentioned earlier about controlling the amount of food coming from the food tank.

Table 1, **Specifications**

	ADULT DOG SIZE DRY FOOD FEEDING AMOUNT (CUPS)	
Portion Control	3-12 lbs.	1/3 to 1 cup
	13-20 lbs.	1 to 1 1/3 cups
	21-35 lbs.	1 1/3 to 2 cups
	26-50 lbs.	2 to 2 2/3 cups
	51-75 lbs.	2 2/3 to 3 1/3 cups
	76-100 lbs.	3 1/3 to 4 ¼ cups
	Over 100 lbs.	4 ¼ cups plus ¼ cup for each 10 lbs. of body weight over 100 lbs.
Wireless control	Wireless connection up to 30 feet to internet	
Dimensions	2 feet in height x 2 feet in width x 2 feet in length	
Power consumption	5V - 12V (estimated), 1W-3W (estimated)	
*Low food reminder	When the food is about ¼ full send reminder	
*Manual dispensing	Drops food within 1 minute of command	
Scale	5% error	
*Setting Timer	Dispenses with 30 seconds of set time	
LEDs	LED Indicator Level: (1: Green, Full) ⇒ (2: Yellow,(¼) Low) ⇒ (3: Red, Empty)	
Weight	2-3 lbs without food	
Container size	Should hold 4 lbs of food	

* Means using for demonstration

Here in *Table 1*, we have the specifications that we expect our auto pet feeder to be able to do.

Ideally, we would like to have wireless control from a remote location with the design having up to 30 feet of wireless connection to WiFi. For portability, we want the design to be compact and lightweight for the average user. The container size fits in with this as it decides the overall loading capacity thus affecting the design of the weight and dimensions. For food reminders, this will be tied in with the scale and will notify the user by LED, display or by the app. Most of the power consumption will be based on a per use basis. This is because the device will only need to draw power when it is being used, otherwise, it is in a low power mode that waits for use. The setting timer and manual dispenser will be able to be set from the app or directly on the design itself.

2.2 Block Diagram

Figure 1, **Block Diagram**
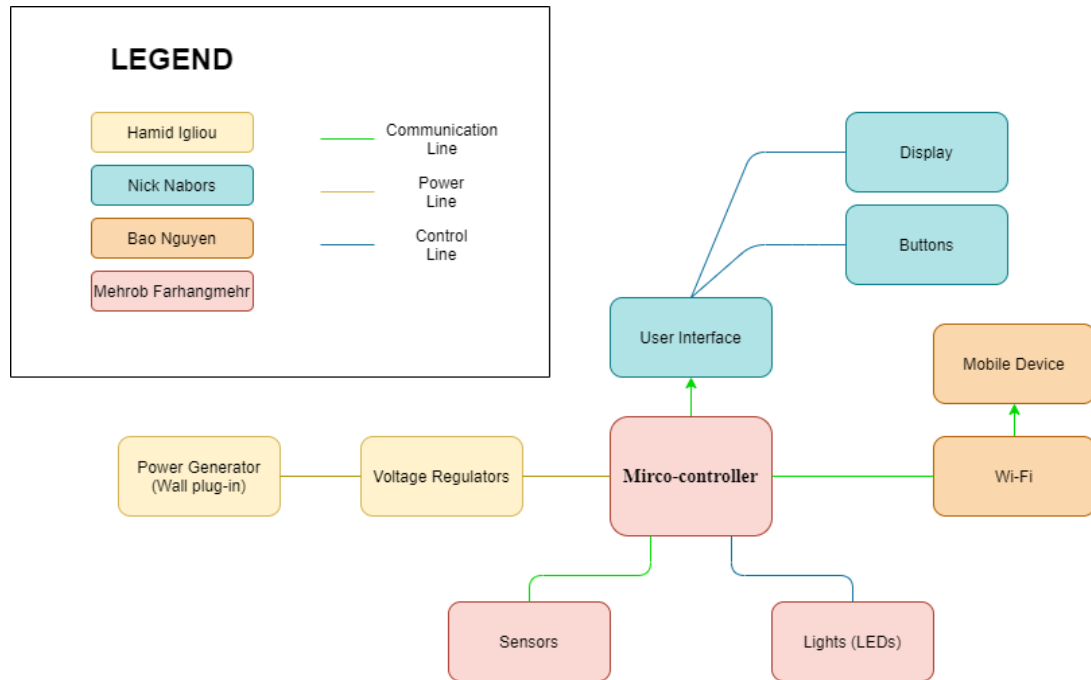
Here in *Figure 1*, We can see that there are many major parts needed for our prototype of the automatic pet feeder. At the center of the Block diagram there is a micro-controller that will be the mastermind and control between all of the parts. Right below the micro-controller there are the Sensors and Lights(LEDs) we can see based on the color of the lines connecting them the sensors will be communicating to the microcontroller and the LEDs will be controlled by MCU. To the left of the MCU there is the power aspect of the project, we can tell by the dark yellow lines that are connecting the "Power Generator" block and the "Voltage Regulator" block to the MCU. Above the MCU we can see there is an User Interface which will communicate to the MCU, and control the Display and Button. Finally on the right side we can see that there is going to be a Wifi module that allows us to connect to a mobile device. This will allow the user to remotely control the Auto pet feeder.

From the color scheme blocks that we have we can see who oversees which part of the project. Hamid is in charge of the yellow blocks, Mehrob is the red block, Nick is the blue, and Bao the orange. As we see in *Figure 1*, there are many moving parts to this product, but we have divided up the parts so we can cover more ground quickly since there are time constraints for our project.

Even though the tasks are split up into parts we will have to work collectively as a team on each section to make sure that everything is moving smoothly and is working properly. The chart only shows who is responsible for each part and are their main focus for the over project.

2.3 Project Milestones (Initial)

Table 2, **Senior Design 1 Milestones (Tentative)**

Week number (Date range of Week)	Milestones
Week 1 (1/11/21 – 1/17/21)	Create Group
Week 2 (1/18/21 – 1/24/21)	Think of Design Idea/Problem
Week 3 (1/25/21 – 1/31/21)	Plan Idea/ Finish Divide and Conquer 1.0
Week 4 (2/1/21 – 2/7/21)	Research
Week 5 (2/8/21 – 2/14/21)	Research, Finish Divide and Conquer 2.0
Week 6 (2/15/21 – 2/21/21)	R&D
Week 7 (2/22/21 – 2/28/21)	R&D
Week 8 (3/1/21 – 3/7/21)	R&D
Week 9 (3/8/21 – 3/14/21)	R&D
Week 10 (3/15/21 – 3/21/21)	R&D
Week 11 (3/22/21 – 3/28/21)	R&D
Week 12 (3/29/21 – 4/4/21)	60 page Draft Senior Design I Documentation
Week 13 (4/5/21 – 4/11/21)	R&D
Week 14 (4/12/21 – 4/18/21)	100 page report submission_updated
Week 15 (4/19/21 – 4/25/21)	R&D
Week 16 (4/26/21 – 5/2/21)	Final Document Due

Table 2 shows the tentative schedule for the semester of Senior Design 1. The main focus of this semester is preparation and research for Senior Design 2 the following semester. The administrative work in this semester will set us up for success in the following semester when we implement our research and planned budget the way we planned in Senior Design 1. This is subject to change on a weekly basis, after we have had our weekly meeting. Even more so if we meet more than once per week to discuss any changes or adjustments that need to be made to help us achieve our goal of success.

<div align="center">

Table 3, **Senior Design 2 Milestones (Tentative)**

</div>

Week number (Date range of Week)	Milestones
Week 1 (5/17/21 – 5/23/21)	Acquiring Materials
Week 2 (5/24/21 – 5/30/21)	Acquiring Materials
Week 3 (5/31/21 – 6/6/21)	Build Prototype
Week 4 (6/7/21 – 6/13/21)	Build Prototype
Week 5 (6/14/21 – 6/20/21)	Test
Week 6 (6/21/21 – 6/27/21)	Build Prototype
Week 7 (6/28/21 – 7/4/21)	Build Prototype
Week 8 (7/5/21 – 7/11/21)	Test
Week 9 (7/12/21 – 7/18/21)	Build Prototype
Week 10 (7/19/21 – 7/25/21)	Build Prototype
Week 11 (7/26/21 – 8/1/21)	Test
Week 12 (8/2/21 – 8/8/21)	Finalize Prototype
Week 13 (8/9/21 – 8/15/21)	Prepare Final Reports
Week 14 (8/16/21 – 8/22/21)	Prepare Final Reports

Table 3 demonstrates the same schedule type of layout that *Table 2* presents. The difference this semester, for Senior Design II, is that we begin building from our research and budget. The administrative work laid out in *Table 3* will help us reach our end goal of having a successful and completed project.

During the Senior Design II semester, as stated in *Table 3*, we will begin with acquiring the materials we have specified. From there, we will begin testing and building of a prototype over the course of Senior Design II. We aim to finish towards the tail end of the semester so we can finalize reports and anything else we need.

Figure 2, **House of Quality**

		Dispensing Weight (Accuracy)	Light Weight	Efficiency (Power Consumed)	Ease of use	Dimensions	Cost
		+	+	-	+	-	-
1) Portable	+		↑↑		↑	↑	↓
2) Low Power	+			↑↑			↑
3) Easy Installation	+				↑↑		
4) Cost	-	↓↓	↓	↓	↓	↓	↑↑
Engineering Target Specifications		5% error	Weighs < 3lbs (Without food)	1 - 3 Watts	Ready to use in < 10 minutes	2 ft x 2ft x 2 ft	<$300
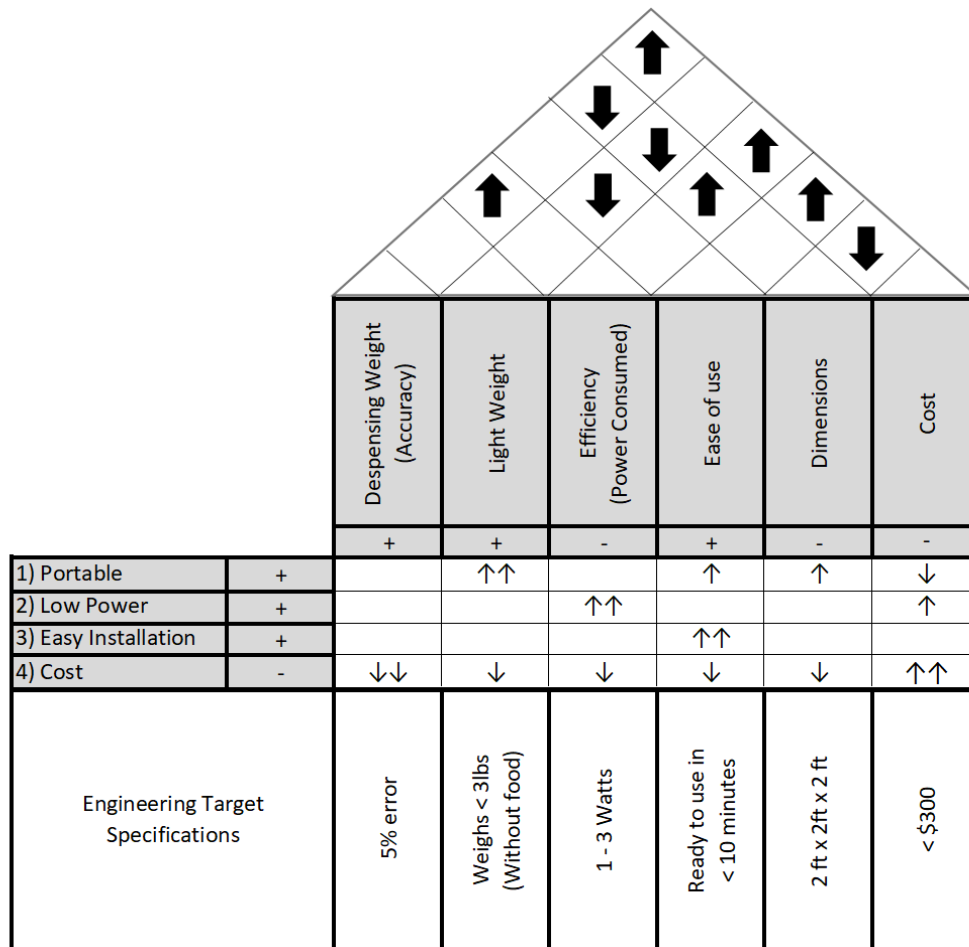
We built a house of quality in order to clarify some of the distinct features of our project as compared to similar products. Although we are not exactly sure how much the pet feeder will cost, we are certain it will not exceed $300 which will make it affordable for every pet owner.

Other qualities include efficiency; our product will be very efficient when it comes to power consumption as we will use low power components, thus our product will not be a financial burden on the owner.

Our pet feeder will also be efficient when it comes to the amount of food dispensed, the use of the scale will give the pet owner the ability to to provide the exact amount of food to their pet. Our feeder will also be easy to use for any owner and the installation also be very simple and wouldn't require any expertise. All these qualities will make our pet feeder very competitive in the market.

3. Research

Will cover all parts that have been considered for the project and selected parts.

3.1 User Interface and Control

The interface and control portions of the design focus on user interaction as well as overall control of the design through user input. The user interface is important to the overall design as it is made to help the user operate without any knowledge of the inner workings of the design.

For the user interface, it is important that it must be easy to navigate for the user to successfully and easily operate the design. This would have to keep a focus on simplicity and efficiency. This can be done in many ways and will be further explored in this section of the design paper.

For the design control, it must be noted that the control for the design must be quick and responsive as well as simple. This carries off of the user interface because the user will be using the interface which directly controls the design through the hardware and software. This will also be further elaborated on throughout this section.

User Interface and Control General Breakdown:
- LCD Display/Menu
  - User Controlled
    - Selection of portion size: small, medium, large (range of dispensing)
    - Scheduling of dispensing
    - Dispensing of food manually
  - Vitals
    - Status of food level in tank (checked after dispensing)
    - Wifi connection status
    - Power level
    - Portion weight calibration
- Button Control
  - Up/Down buttons
  - Select button

The user will scroll through the menu to access or view any features that are stated above in the user interface and control breakdown.

The breakdown describes the basic layout for the menu that the user will access. The user controlled section is what the user can actually modify by control command. The vitals section will display the listed vitals to the user to show the various processes of the design that may be important. There will be a numpad to enter weights for the portion size calibration and the up/down buttons will navigate the menu and use the select button to use the selected menu option.

OSEPP LCD Module
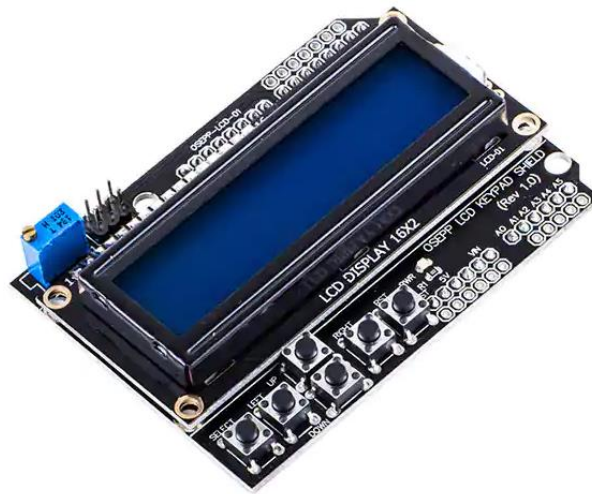
Figure 3, **OSEPP Electronics LCD and Button Module**

The 16X2SHD-01 LCD Module uses the SPLC780D LCD display to display the desired results. This module has buttons, an LCD display, a driver/controller for the LCD and is Arduino compatible.

The 16X2SHD-01 LCD Module has the following basic features and specifications:
- Compatible with the Arduino ecosystem
- Can display up to 16 characters on two lines each
- Integrates the SPLC780D LCD controller and driver
- Operates at 5 V
- Comes with a 5 button keypad: select, up, down, left, right
- Has a reset button that is compatible with the Arduino ecosystem
- Approximately ~$14.99 market value

Figure 4, **LCD Button Schematic**
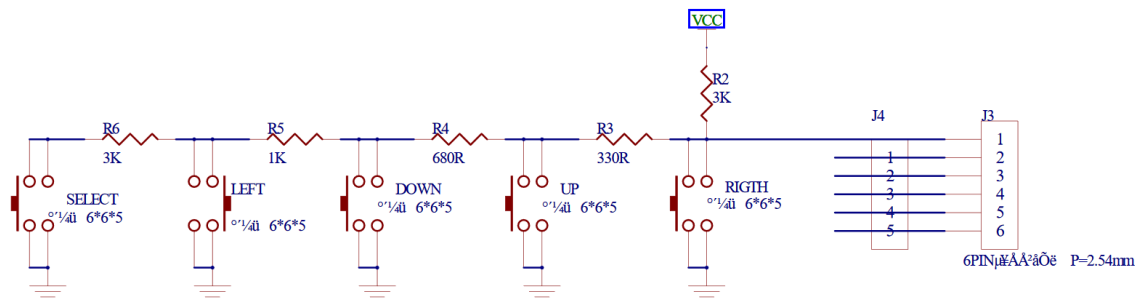
Figure 4 is the schematic for the buttons of the LCD driver/control board of the 16X2SHD-01 LCD Module. It is set up in a series fashion and is directly integrated into the board without any further need to attach any extra components. This helps provide a cheap method of control that will be easy to integrate into our Arduino ecosystem and will help us control the functions of the system as a whole.

Figure 5 is the schematic for the LCD itself.

Many of the segments are connected through the means of parallel wire structures that correspond to the specific function and display.

You can see the Vcc power with an LED indicator that is directed to a saturated BJT transistor meaning it is being used as a switch.

The variable resistor RP1 is being used as a dimming control for the LCD screen brightness. RST is the reset button that has two parallel routes to different modules of the LCD main module. There are multiple parts to this schematic that aren't presented in detail in the datasheet but are not necessary to the overall outcome to the final design to be used.
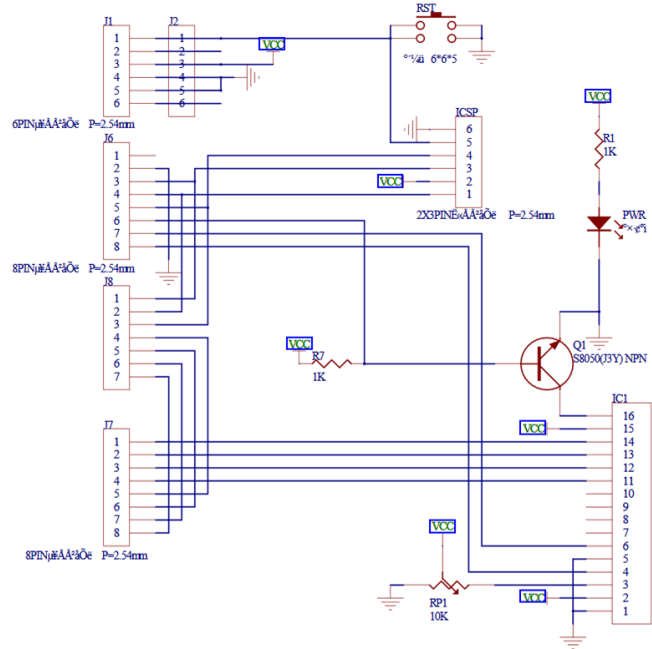
Figure 5, **LCD Schematic**

Figure 6 is the block diagram for the LCD controller, SPLC780D, and the driver, Sunplus. In the block diagram, our power supplies are VDD and VSS which will be +5V and 0V or GND, respectively. Since this is compatible with the Arduino system, we can use the generated clock signal from the MCU and the MCU will send the necessary data signals to the data pins of the LCD module, as shown in the block diagram.

The control buttons on the LCD module are already implemented in the design with the block diagram but it is not shown in this instance. The control buttons will be able to control and send signals to the chip to be processed. This will be explained later.

Figure 6, **SPLC780D Block Diagram**
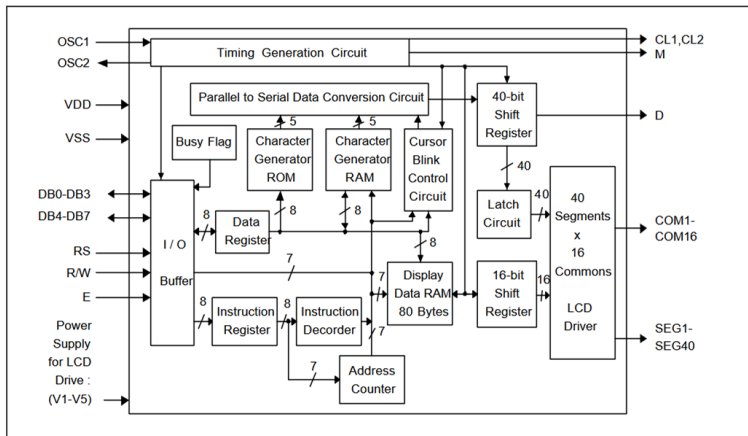
Power supply for the LCD drive will be implemented to make sure the LCD segments can display the proper data for the design user to see any important information that is desired. This could be the status of the container, for example.

Figure 7, **SPLC780D Signal Descriptions**

Mnemonic	PIN No.	Type	Description
VDD	33	I	Power input
VSS	23	I	Ground
OSC1	24	-	Both OSC1 and OSC2 are connected to resistor for internal oscillator circuit. For
OSC2	25		external clock operation, the clock is input to OSC1.
V1 - V5	26 - 30	I	Supply voltage for LCD driving.
E	38	I	A start signal for reading or writing data.
R/W	37	I	A signal for selecting read or write actions.
			1: Read, 0: Write.
RS	36	I	A signal for selecting registers.
			1: Data Register (for read and write)
			0: Instruction Register (for write),
			Busy flag - Address Counter (for read).
DB0 - DB3	39 - 42	I/O	Low 4 -bit data
DB4 - DB7	43 - 46	I/O	High 4-bit data
CL1	31	O	Clock to latch serial data D.
CL2	32	O	Clock to shift serial data D.
M	34	O	Switch signal to convert LCD waveform to AC.
D	35	O	Sends character pattern data corresponding to each common signal serially.
			1: Selection, 0: Non-selection.
SEG1 - SEG22	22 - 1	O	Segment signals for LCD.
SEG23 - SEG40	80 - 63		
COM1 - COM16	47 - 62	O	Common signals for LCD.

To control our design, the buttons interface for the user will impact the pinouts in *Figure 7*. As described in the block diagram, our power source will be VDD and +5V from our regulated voltage. VSS will be looped in with the grounding of our PCB from the regulated power source.

OCS1 will use an external clock from the Arduino module to generate the LCD dot matrix data to be presented to the LCD segments. Low power mode for this design will not be necessary as the design will use a direct current regulated input from an alternating current source. Thus, we can use the maximum clock speed that is necessary for the LCD to operate at standard specifications.

V1-V5 are the supply voltages for driving the LCD and must be set up properly to ensure overall success. Though, it may not be necessary as it is already set up in the LCD module itself, it is safe to understand how this works.

E or enable will allow the Arduino to use and communicate with the LCD module. R/W will set the signal to read or write based on the data communicated from the MCU. RS will select the necessary registers to carry out the actions needed to run the display.

The data bits pins will take the data sent from the MCU to be processed and ultimately displayed by the process of the LCD driver as shown in the block diagram previously. The segment signals and the common signals are important for the communication between the LCD driver as well as the LCD itself. It is necessary to understand how these two devices communicate even though they are already set up together in this module workflow.

The diagram in *Figure 8* shows the character combinations for the various outputs from the Arduino MCU to the OSEPP LCD module.

The combinations are paired bits in groups of four to give enough variety of combinations to have on display.

The chart splits the eight bits into two four bit rows and columns. The higher four bits control the columns and the lower four bits control the rows. The columns, as shown, house different categories of the character styles

Figure 8, **Character Code Combinations**

that are present. This is obvious from column three, for example. All the numbers are in the third column and so on.

It is not expected that the design to be implemented will use all the characters on the chart list presented here. But it doesn't hurt to understand that the design can implement more characters, if needed, to display any sensor data that may need to be read to the LCD screen for the user to gather information from.

It is helpful that these characters are already in the code for our design to implement as it would be very difficult and time consuming to create characters for the LCD screen that the design would need to have available.

Later, in this section, the actual process of this procedure is shown and how it is implemented and it will be elaborated on.
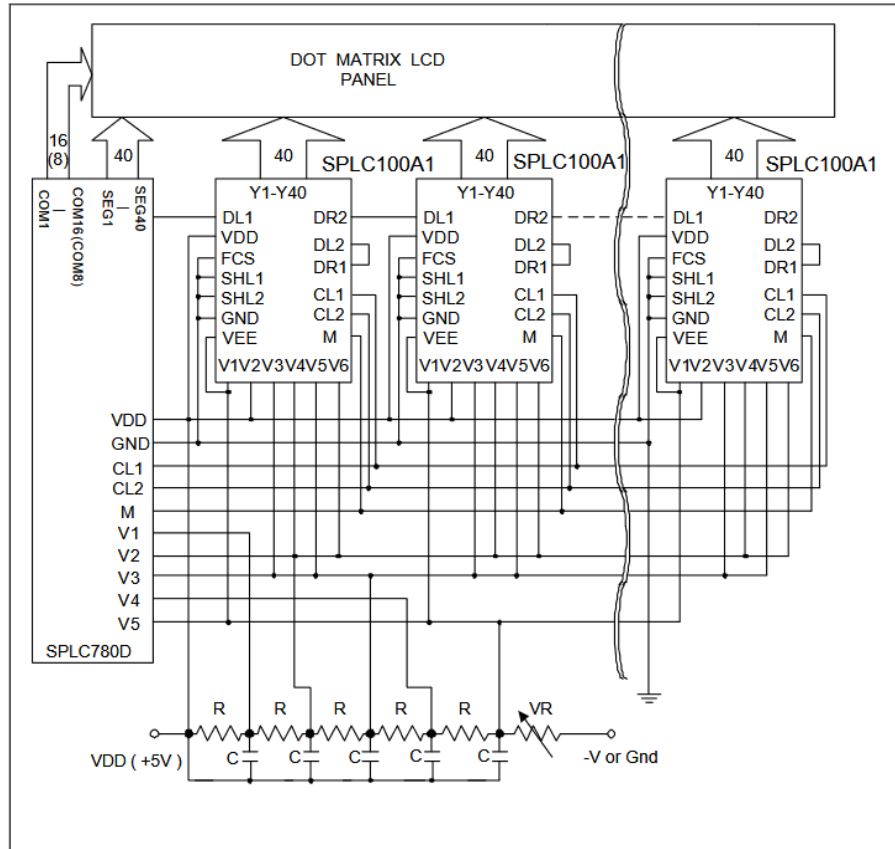
Figure 9, **SPLC780D Application Circuit**

The schematic presented in *Figure 9* shows the interaction between various components of the LCD module.

The bottom row of the circuit diagram shows the power control for the various LCD driver settings. The +5V that comes in on the LCD module is the power that the design will have coming in from the regulated alternating current source to direct current source. It is divided up through a resistor-capacitor network that sends the required amount of power to the LCD driver segment power pins.

The SPLC780D driver on the left of the diagram is the main control for the LCD module system. It is the main control of the whole display system and processes the signals sent from the Arduino MCU and outputs them to the dot matrix LCD panel in the top part of the diagram.

The SPLC100A1 devices span across the LCD panel as well and there are multiple that control the power and display for specific segments of the display. They use the same regulated direct current power source that is being implemented in our design. The necessary code for the LCD Module is already compiled on the SPLC780D. Our code will be implemented in the Arduino and will be compiled on the Arduino but it will interact with the LCD module through the presented data in the connected pins of the LCD module.

14
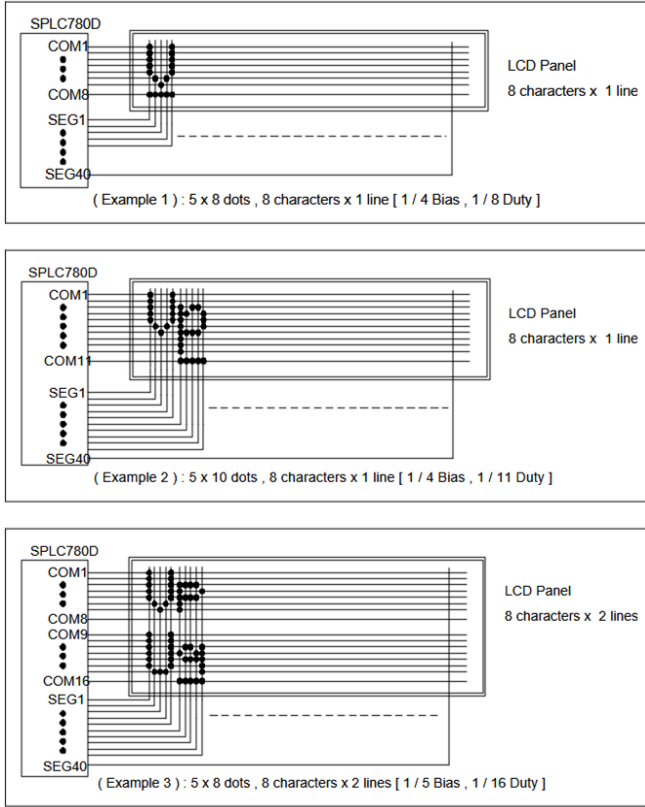
Figure 10, **SPLC780D LCD Applications**

Figure 10 on the left shows the process for displaying the characters on the LCD module screen.

It is being controlled by the SPLC780D to display the small amount of pixels needed to display characters on the screen. In example one, on the left, the cursor is shown to begin at the first index of the LCD dot matrix. Once the character is written to the LCD the SPLC780D moves the cursor to the next index and repeats the process. Example one and two show the process for eight characters on one line and how they are indexed.

Example three, in the image on the left, shows the same previous process for the eight characters but on two lines instead of one. The overall process is carried out the same way and can be done simultaneously as the lines use different com and seg wires to display the characters.

In the next example, example four on the right, *Figure 11,* a similar process is shown for the characters but for sixteen characters on one line instead of eight. The process is overall the same even with the change in amount of characters.

The final example presented is example five and it shows how four characters are presented on two lines. Overall, the process for most of these examples doesn't change but just uses varying signal and wire combinations to carry out the necessary display of characters. This is simple and can be easily implemented into our design.
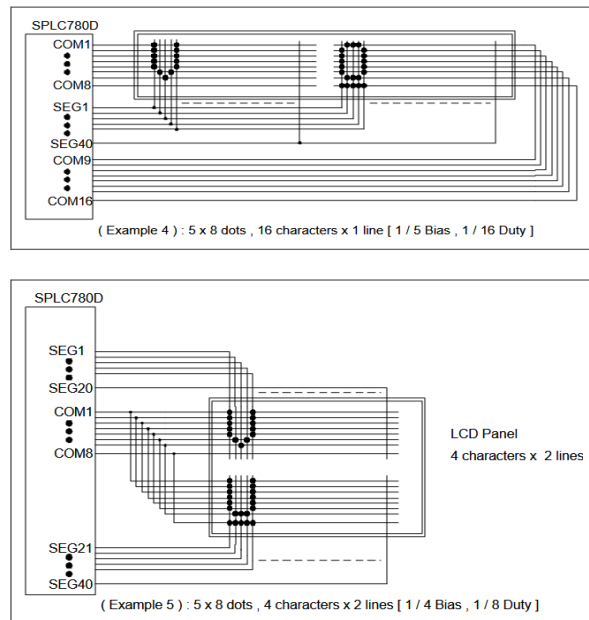
Figure 11, **SPLC780D LCD Applications, cont.**

15

Figure 12, **SPLC780D Character Generator ROM**

Figure 12 at the top is for the character generation ROM which can generate different character sets as shown in the image above.

Though it may not be necessary to implement this feature for the purpose of the design, it is interesting to possibly propose in future possible designs that can implement different languages as well as characters or outputs.

Having multiple localizations could be beneficial for commercial purposes of a design like this and could help promote the design to a broader user base. The character generator ROM can be a useful feature to show varying errors that can be presented in a form of confusion but that will not be implemented in the final design.

Even though some of the features presented in the SPLC780D may not be used for our final design, it is worthy to note that some of these features could be implemented in future designs or sooner, if time permits, and can be implemented based on the presented material within the SPLC780D datasheet. The large amount of presentation features do give more creative space for the design of the user interface and control menu.

In *Figure 13* below, the instruction process for the SPLC780D is shown. This will be the process carried out for the eight digit display spanning one line.

The process shown carries through steps beginning with powering on the LCD module display. Again, the design uses the varying power from the designs alternating current to direct current +5V regulated source. It is then further divided, as necessary, for the specific LCD display segments and carries out the instructions embedded in the SPLC780D until the final instruction is carried out. The characters are written to the LCD display from left to right as shown in *Figure 13*.

Though, we do not implement this in our actual design process as it is already completed for use in our design, it is important to acknowledge and understand the processes that are running in the hardware that the design will be implementing in the embedded software. The instructions below may need to be referenced at a later time during the implementation of the design or, at the minimum, be understood to fix any possible errors that may arise in the implementation of the design process.

Figure 13, **SPLC780D 8-Digit 1-Line Display**

No.	Instruction	Display	Operation
1	Power on. (SPLC780D starts initializing)		Power on reset. No display.
2	Function set RS R/W DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0 0 0 0 0 1 1 0 0 X X		Set to 8-bit operation and select 1-line display line and character font.
3	Display on / off control 0 0 0 0 0 0 1 1 1 0	_	Display on. Cursor appear.
4	Entry mode set 0 0 0 0 0 0 0 1 1 0	_	Increase address by one. It will shift the cursor to the right when writing to the DD RAM/CG RAM. Now the display has no shift.
5	Write data to CG RAM / DD RAM 1 0 0 1 0 1 0 1 1 1	W_	Write " W ". The cursor is incremented by one and shifted to the right.
6	Write data to CG RAM / DD RAM 1 0 0 1 0 0 0 1 0 1	WE_	Write " E ". The cursor is incremented by one and shifted to the right.
7	:	:	
8	Write data to CG RAM / DD RAM 1 0 0 1 0 0 0 1 0 1	WELCOME_	Write " E ". The cursor is incremented by one and shifted to the right.
9	Entry mode set 0 0 0 0 0 0 0 1 1 1	WELCOME_	Set mode for display shift when writing
10	Write data to CG RAM / DD RAM 1 0 0 0 1 0 0 0 0 0	ELCOME _	Write " "(space). The cursor is incremented by one and shifted to the right.
11	Write data to CG RAM / DD RAM 1 0 0 1 0 0 0 0 1 1	LCOME C_	Write " C ". The cursor is incremented by one and shifted to the right.
12	:	:	
13	Write data to CG RAM / DD RAM 1 0 0 1 0 1 1 0 0 1	COMPAMY_	Write " Y ". The cursor is incremented by one and shifted to the right.
14	Cursor or display shift 0 0 0 0 0 1 0 0 X X	COMPAMY_	Only shift the cursor's position to the left (Y).
15	Cursor or display shift 0 0 0 0 0 1 0 0 X X	COMPAMY_	Only shift the cursor's position to the left (M).
16	Write data to CG RAM / DD RAM 1 0 0 1 0 0 1 1 1 0	OMPANY_	Write " N ". The display moves to the left.
17	Cursor or display shift 0 0 0 0 0 1 1 1 X X	COMPAMY_	Shift the display and the cursor's position to the right.
18	Cursor or display shift 0 0 0 0 0 1 0 1 X X	OMPANY_	Shift the display and the cursor's position to the right.
19	Write data to CG RAM / DD RAM 1 0 0 1 0 0 0 0 0 0	COMPAMY_	Write " " (space). The cursor is incremented by one and shifted to the right.
20	:	:	:
21	Return home 0 0 0 0 0 0 0 0 1 0	WELCOME_	Both the display and the cursor return to the original position (address 0).

No.	Instruction	Display	Operation
1	Power on. (SPLC780D starts initializing)		Power on reset. No display.
2	Function set RS R/W DB7 DB6 DB5 DB4 0 0 0 0 1 0		Set to 4-bit operation.
3	0 0 0 0 1 0 0 0 0 0 X X		Set to 4-bit operation and select 1-line display line and character font.
4	0 0 0 0 0 0 0 0 1 1 1 0	_	Display on. Cursor appears.
5	0 0 0 0 0 0 0 0 0 1 1 0	_	Increase address by one. It will shift the cursor to the right when writing to the DD RAM / CG RAM. Now the display has no shift.
6	1 0 0 1 0 1 1 0 0 1 1 1	W_	Write " W ". The cursor is incremented by one and shifted to the right.

Figure 14, **SPLC780D 8-Digit 1-Line Display**

In *Figure 14* above, the same process from *Figure 13* is shown in the image. Again, the +5V regulated alternating current to direct current source is used and further divided to the LCD to present the written data.

The main difference with this process is that the operations or instructions use 4-bit operations rather than 8-bit in *Figure 13*. This figure, *Figure 14,* also shows eight digits for a one line display.

Figure 15, is a similar process carried out from *Figure 13* and *Figure 14* but the variation here is for an eight digit LCD display for two lines rather than one line.

When the design is being implemented, it may not require the use of the instructions presented in these figures but it can be possibly important later in the design as an issue could arise and need attention related to these given instructions.

No.	Instruction	Display	Operation
1	Power on. (SPLC780D starts initializing)		Power on reset. No display.
2	Function set RS R/W DB7 DB6 DB5 DB4 DB3 DB2 DB1 DB0 0 0 0 0 1 1 1 0 X X		Set to 8-bit operation and select 2-line display line and 5 x 8 dot character font.
3	Display on / off control 0 0 0 0 0 0 1 1 1 0		Display on. Cursor appear.
4	Entry mode set 0 0 0 0 0 0 0 1 1 0		Increase address by one. It will shift the cursor to the right when writing to the DD RAM / CG RAM. Now the display has no shift.
5	Write data to CG RAM / DD RAM 1 0 0 1 0 1 0 1 1 1	W_	Write " W ". The cursor is incremented by one and shifted to the right.
6	:	:	:
7	Write data to CG RAM / DD RAM 1 0 0 1 0 0 0 1 0 1	WELCOME_	Write " E ". The cursor is incremented by one and shifted to the right.
8	Set DD RAM address 0 0 1 1 0 0 0 0 0 0	WELCOME	It sets DD RAM's address. The cursor is moved to the beginning position of the 2nd line.
9	Write data to CG RAM / DD RAM 1 0 0 1 0 1 0 1 0 0	WELCOME T_	Write " T ". The cursor is incremented by one and shifted to the right.
10	:	:	:
11	Write data to CG RAM / DD RAM 1 0 0 1 0 1 0 1 0 0	WELCOME TO PART_	Write " T ". The cursor is incremented by one and shifted to the right.
12	Entry mode set 0 0 0 0 0 0 0 1 1 1	WELCOME TO PART_	When writing, it sets mode for the display shift.
13	Write data to CG RAM / DD RAM 1 0 0 1 0 1 1 0 0 1	ELCOME O PARTY_	Write " Y ". The cursor is incremented by one and shifted to the right.
14	:	:	:
15	Return home 0 0 0 0 0 0 0 0 1 0	WELCOME TO PARTY	Both the display and the cursor return to the original position (address 0).

Figure 15, **SPLC780D 8-Digit 2-Line Display**

Figure 16, **LCD Module Rear Pin View**

In *Figure 16*, the rear view of the LCD module is shown. The pins on the rear of the LCD module correspond to the pins on the front of the module.

In the design to be implemented, the pins will be inserted into a connector which will run wire from the LCD module to the corresponding pins on the Arduino MCU.
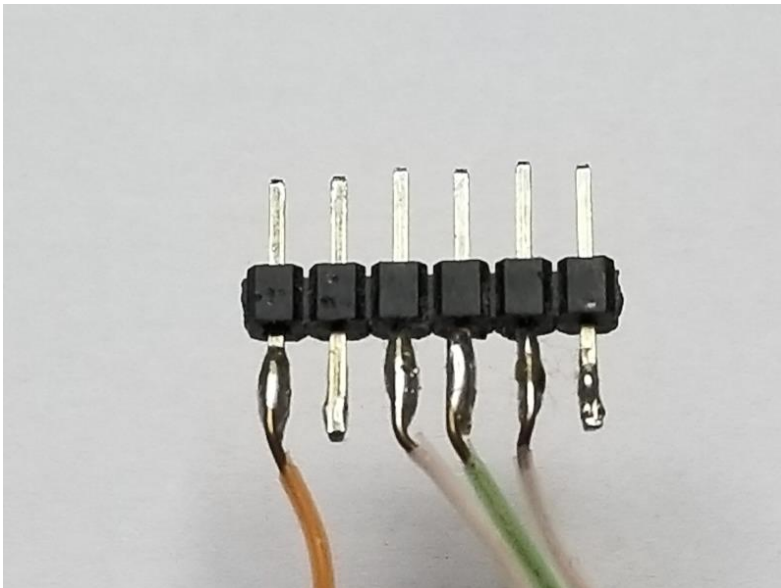
In *Figure 17*, the process for soldering the rear pins from the LCD module in *Figure 16* to the PCB pins of the Arduino MCU.

The soldering method has been chosen rather than using pin connectors as soldering can be more sturdy and present better resistance to outside forces such as bumps or drops of the design. This will improve the overall durability of the design.

Figure 17, **Basic Solder Connections**

```
/*
  LCDKeypad.cpp
*/
#if ARDUINO >= 100
  #include "Arduino.h"
#else
  #include "WProgram.h"
#endif


// include this library's description file

#include <LiquidCrystal.h>
#include "LCDKeypad.h"

LCDKeypad::LCDKeypad() : LiquidCrystal(8, 9, 4, 5, 6, 7)
{
}

int LCDKeypad::button()
{
  static int NUM_KEYS=5;
  static int adc_key_val[5] ={
    30, 150, 360, 535, 760     };
  int k, input;
  input=analogRead(0);
  for (k = 0; k < NUM_KEYS; k++)
  {
    if (input < adc_key_val[k])
    {
      return k;
    }
  }
  if (k >= NUM_KEYS)
    k = -1;      // No valid key pressed
  return k;
}
```

Figure 18, **LCD Sample Code**

Figure 18 shows sample code provided by OSEPP for the Arduino interface for the LCD module control.

The sample code given was written in C++ which can be compiled in the Arduino software environment.

The Arduino header file, "Arduino.h" is a standard library provided by Arduino to help embedded systems interface with the MCU and it must be imported as shown in the header section of the code sample. "LiquidCrystal.h" as well as "LCDKeypad.h" must also be imported to the project as shown in the code sample.

The first function is written using the scope resolution operator which defines a function inside of an outside class. In this sample code, the first function is the constructor for Liquid Crystal which takes in the values 4-9 as parameters.

The second function returns the button that is pressed. The number of keys and the key values from the analog to digital converter are initialized as an integer and integer array, respectively. The input integer is declared and initialized to the Arduino function of analogRead(0). This function, when connected to pin 0, will read the data from the LDC module button press and run it through the analog to digital converter in the Aruduino to convert it to the correct value.

The for loop runs through the number of keys and compares the analog to digital converter key value to the analog to digital converter input value and returns the value to the Arduino to be displayed.

If the compared analog to digital values are greater than the number of keys, then the Ardunio will determine that the key that was pressed is not a valid key based off of the code compiled to the Arduino.

Adafruit LCD Module
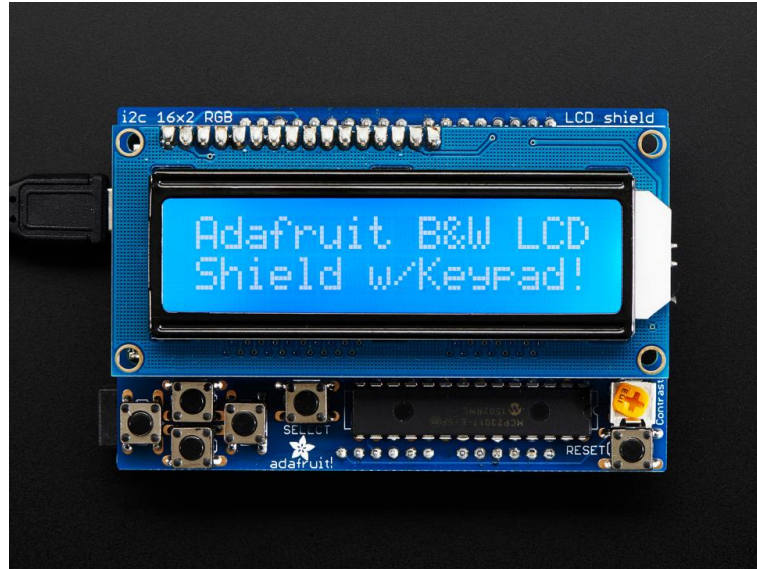
Figure 19, **Adafruit LCD Module**

The Adafruit LCD Module is similar to the OSEPP LCD Module. They both are 16x2 LCD display modules with button controls directly attached to their shield components. As with such similarities, the functionalities are fundamentally similar. They both have similar libraries for software when it comes to programming the button control as well as the LCD output display control from the written and embedded code in the Arduino device.

It is noted later in this document comparison why the OSEPP LCD Module was chosen over this Adafruit LCD Module. For a comparison between these two parts, the main reasoning for selecting the OSEPP LCD Module came down to the price comparison between the two parts. There is a five dollar difference in the price between the two parts mainly due to the name-brand functionality of the Adafruit product suite.

The Adafruit LCD Module is a good alternative to the OSEPP LCD Module as they are very similar and Adafruit gives its customers great documentation and tutorials on how to set up their products, especially with Arduino or compatible microcontroller systems. It will also operate and set up similarly to the OSEPP LCD Module which is the main reason this part is listed as an alternative display for the overall design.

Another comparison that was a determining factor in choosing the OSEPP LCD Module that has no large argument with this part because they are similar in operation and setup is the power consumption comparison. Both parts operate at similar voltages and current draws that had no effect on the ultimate part comparison.

For this part, the determining factor really just came down to the price of the OSEPP LCD Module over the price of this part.

Nokia 5110/3310 Monochrome LCD

Figure 20, **Nokia LCD**

The Nokia 5110/3310 LCD is a popular display in the Arduino compatible display market. It's been noted for its easy to setup and easy to program model. The reason that this display was not chosen over the OSEPP LCD Module comes down to a few factors that will be further elaborated on on this page.

With the low price of this part, about five dollars less, when compared to the OSEPP LCD Module, it still is less desirable of a choice for this project design as a few factors seem to conflict with the overall part picking choices.

This device has a lower power consumption when compared to the OSEPP LCD Module and can display more on its screen compared to the 16x2 display of the OSEPP LCD Module. It does not have a backlit display which does help with the power consumption over long periods of use.

This LCD also has a lot of documentation and support on the internet which does make it a viable option but not our first when it comes down to the actual design that we must implement. The previous two parts, OSEPP LCD Module and the Adafruit LCD Module, both have buttons integrated directly into their shields or boards which is great for our design as we will not have to find a separate part for the control of the menu which will be displayed on the LCD.

This part is still in for consideration though as the community and support for the setup and use is very strong. Not having controls is what was really the main consideration for not using this part for the final design of our project for the control of the menu from the Arduino.

SSD1306 OLED Display

Figure 21, **Adafruit OLED Display**

The Adafruit OLED Display is another proposed option over the OSEPP LCD Module. It does have similar characteristics and determining factors just like the previous part, the Nokia LCD. OLED displays have been receiving notoriety and popularity in recent years as they are being implemented in phone screens due to their low power consumption and high color contrast.

This part, when compared to the OSEPP LCD Module, is a little more pricey. The price isn't the only determining factor for choosing the OSEPP LCD Module as there are other factors that dissuade our final decision in picking this part. But this part is an interesting pick for the purpose that it does have good documentation and support from the community and from Adafruit.

Another factor that the OSEPP LCD Module has an advantage of the Adafruit OLED display would be that it consumes less power over time than this part does. Even with the boasted low-power from OLED displays, it uses more power mainly due to the factors of the greater possibilities of display ranging from graphics to text versus just two lines of text on the OSEPP LCD Module.

Also, this OLED display does not come with buttons or control which is what is ultimately desirable about the OSEPP LCD Module and the Adafruit LCD Module. But, nonetheless, this display is an interesting alternative to the OSEPP LCD Module as it provides many presentable and eye catching features that would not be fully necessary to implement into our design but could enhance it further.

Adafruit TFT Touch Shield with Resistive Touch Screen

Figure 22, **Adafruit TFT Touch Display**

The final part that was considered for implementation into the final design is an ambitious part. The Adafruit TFT Touch Display would be a very interesting and useful display to implement in our design but was not picked over the OSEPP display for some factors that will be further elaborated on.

Because this display implements touch as well as the large size of the screen, it is more on the pricey side when compared to the OSEPP LCD Module. It is another popular display in the market and compatible with the Arduino or Arduino similar suite and has a lot of support and tutorials from the community.

The power that is consumed from this display is one of the factors that is the reason why we chose the OSEPP LCD Module over this display. Because this display is so large and utilizes the touch screen functionality, it consumes more power than any of the other parts that have been proposed for use.

What made this an interesting alternative and an interesting possible choice, is the factor that it is a touch screen display and would not need to utilize a separate control or separate part for controlling the menu system from the Arduino. But it can be noted that with touch screen functionality, it can be a time consuming and difficult setup to manage when the design calls for simplicity and when time is a factor that plays into the ultimate design of the final implementation.

This design could be a good part in a future implementation of this project design but it is not currently important for the time being when compared with the OSEPP LCD Module. The OSEPP LCD Module is the best choice when it comes down to all the factors of this project design at this early of the design.

Chosen Part: OSEPP LCD Module
The reasoning behind choosing the OSEPP LCD Module comes down to many factors that were taken into consideration. They can be broken up in the following ways:
- Pricing
- Ease of Setup/Use
- Power Usage
- Future Interest

For pricing:
- OSEPP LCD Module
  - $14.95 on Digi-Key
- Adafruit LCD Module
  - $19.95 on the Adafruit website
- Nokia 5110/3310 Monochrome LCD
  - $9.95 on Digi-Key
- SSD1306 OLED Display
  - $17.50 on the Adafruit website
- Adafruit TFT Touch Shield Display
  - $34.95 on the Adafruit website

For ease of setup and use:
- OSEPP LCD Module
  - Good documentation and setup tutorials on website
  - Comes with buttons and control built-in
- Adafruit LCD Module
  - Good documentation and setup tutorials on website
  - Comes with buttons and control built-in
- Nokia 5110/3310 Monochrome LCD
  - Good documentation
  - Requires separate buttons for control and libraries for graphics that will not be utilized for this design
- SSD1306 OLED Display
  - Good documentation
  - Requires separate buttons for control
  - Utilizes libraries for graphics that will not be utilized for this design
- Adafruit TFT Touch Shield Display
  - Good documentation
  - Requires libraries for graphics that will not be utilized for this design and the touch screen functionality is not needed for this early of a design

For power usage:
- OSEPP LCD Module
  - 2 mW, Max
- Adafruit LCD Module
  - ~2 mW, Max
- Nokia 5110/3310 Monochrome LCD
  - 264 mW, Max
- SSD1306 OLED Display
  - 2.5 mW, Max
- Adafruit TFT Touch Shield Display
  - 330 mW, Max

For future interest:
- OSEPP LCD Module
  - Currently the working, implemented part
- Adafruit LCD Module
  - Similar outcome as OSEPP but pricier
- Nokia 5110/3310 Monochrome LCD
  - Interesting old and popular display
- SSD1306 OLED Display
  - Low-power display that can utilize graphics and possibly a prettier future menu system with buttons to control
- Adafruit TFT Touch Shield Display
  - Ideal display with possibility for interactive menu system and other touch feature functionality

After taking these factors into consideration, the OSEPP LCD Module best fits our design at the current stage of development and shows to be the best option from all of the categories above. The main factors taken into consideration would be the price, power consumption and user friendliness of setup and implementation.

If this were to be a future, second implementation with added revisions and upgrades, the Adafruit TFT Touch Shield display would be a great option to utilize to give the user the best interaction with the design and this would go hand-in-hand with their experience of the final product.

A touch screen would be the most desired but the OSEPP LCD Module works best for the current design at hand and fits the budget along with all the necessary requirements needed to carry out the final product design for the first completion. Future revisions or additions would implement more features of a menu system and display for users to fully and comfortably interact with.

3.2 Power supply

Importance of Power Conversion

Today we are living in a new era of industrial revolution. The first industrial revolution started around the end of the 18th century when the steam engine was invented. The mechanical power generated by the engine powered all the industrial machines that revolutionized human life at that time. However, mechanical power had many limitations for example: lower speed, greater losses, inefficient transmission, and pollution.

The second industrial revolution started when the electrical power was utilized for powering industrial machines. Electrical power was clean, efficient, and could be transmitted over long distances. The replacement of mechanical power by electrical power increased efficiency and productivity. However, there were still many problems with electrical power, one of the most challenging problems was the efficient control of electric power. Relay coils and vacuum tubes were used for power control, but these devices were bulky, complex, and required high maintenance. The third industrial revolution started with the invention of semiconductor transistors. This revolutionary discovery led to the creation of computers and information technologies. However, the invention of Silicon also revolutionized the area of electric power control and gave birth to the new field of power electronics.
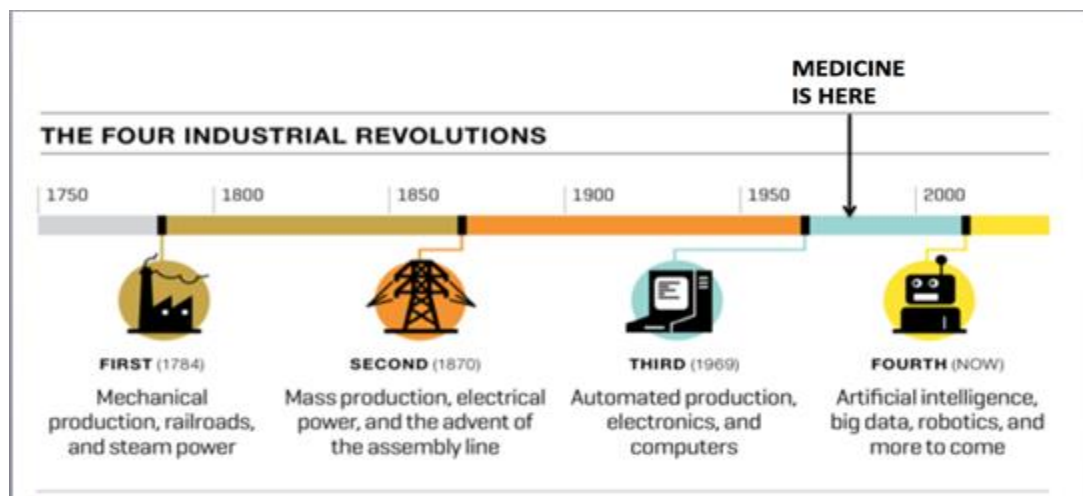
Figure 23, **Timeline of industrial revolution**

Power Electronics

As described earlier, the invention of Silicon based semiconductor devices gave birth to a new field of power electronics. Power electronics can be defined as the use of semiconductor electronic devices for the purpose of power conversion and control. The field of power electronics differs from digital or analog electronics in the respect that power electronic systems are capable of handling large amounts of voltages or currents. Over the

years, a large number of very efficient power electronic devices have been created such as power MOSFETs, IGBTs, GTOs, and power diodes. Using these solid-state devices, power converters can be created which can handle large voltage and current ratings. The field of power electronics is concerned with the design and development of electric power converters. The electric power converters can be divided into four main categories which are as following:

1.   Rectifiers
2.   Inverters
3.   Choppers
4.   AC regulators

It is common that electric power is classified under two main types: AC and DC. In many power conversion systems, there is a need for conversion of electric power from one type to another type depending on the need and requirements of the application. The above categories of power converters are also classified on the basis of type of power conversion.
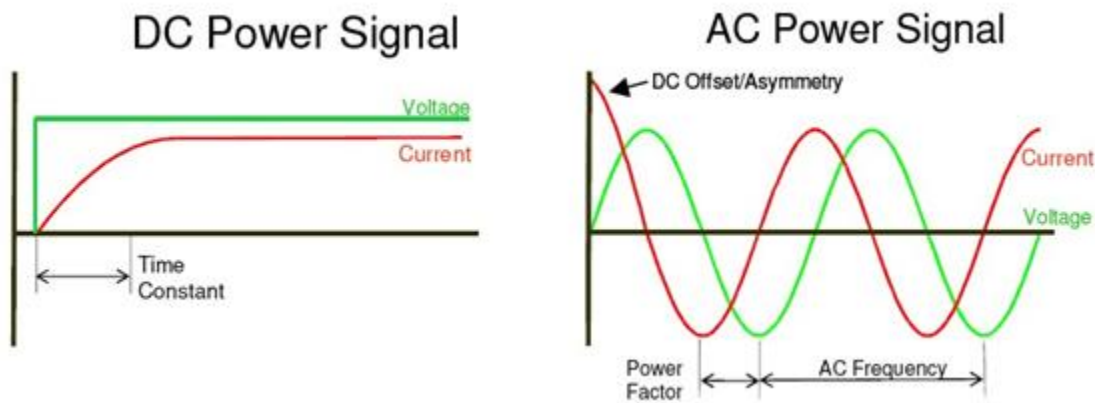
Figure 24, **AC vs DC power signal**

A rectifier is a power electronic converter which converts AC power into DC power. There are many different types and topologies of rectifiers including: single phase, 3 phase, 12 pulse, and SCR rectifiers. An inverter is a power electronic converter which converts DC power into AC power. Inverters are an essential component of battery powered systems such as solar PV and UPS backup systems. There are many different types of inverters such as single phase, three phase, pure sine wave, and impedance source inverters. A DC-DC converter or a chopper is a power electronic converter which converts DC power at one level to DC power at another level. This converter achieves this conversion by switching or chopping a DC signal at high frequency. Choppers are an important element of switch mode power supplies which are used in a large number of electronic appliances and machines. There are many different topologies of DC-DC converters such as fly-back, buck, boost, and SEPIC. An AC voltage regulator is a power electronic converter which converts AC power at one level to AC power at another level. These types of converters are used in motor drives and industrial power control systems.
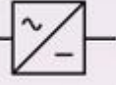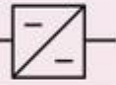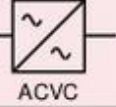
Figure 25, **Classification of power electronic converters**

The purpose is to design and develop a 120VAC to 5VDC/3.3VDC power supply for the pet feeder. The design will be of course based on rectifiers as the conversion from AC to DC is required. In addition to that, the design of the power supply should also incorporate a filtering capacitor and voltage regulator for a smooth output.

Power Supply Design

The first step in the design of AC to DC converter or power supply is to identify all the major components required for the construction of power supply. A list of main components is provided as following:

1. Rectifier
2. Transformer
3. Filter capacitor
4. Voltage regulator
5. Current protection device (fuse)
6. Voltage protection device (varistor)
7. Test equipment

The individual design and detailed description for each of these components is provided in the subsequent subsections.

Reference Design

A reference design for a multi-output power supply is presented in the following figure:
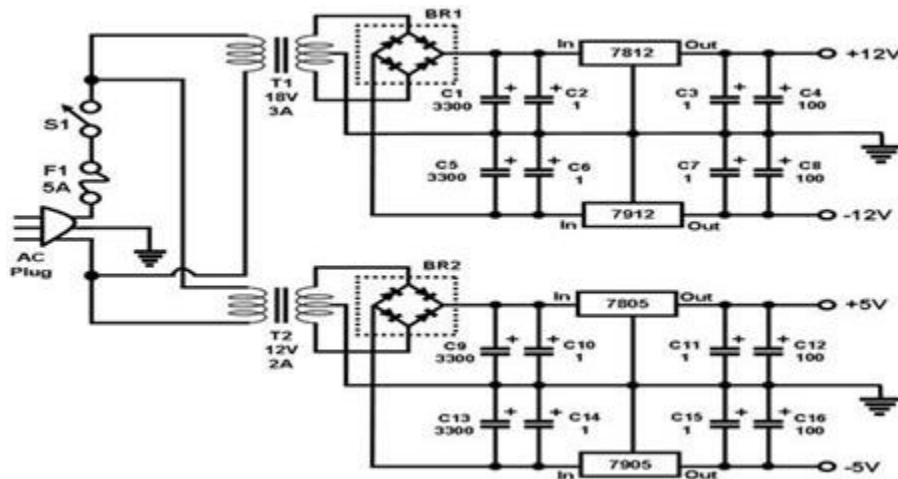
Figure 26, **Multi-level output power supply reference design**

This reference schematic circuit describes a multi-level dual output power supply. The power supply consists of four channels with the following voltage outputs available at the power supply output channels.

1.   +12V – GND
2.   -12V – GND
3.   +5V – GND
4.   -5V – GND

This power supply provides negative as well as positive output voltages; it is known as a dual output power supply since it provides two voltage levels 12V and 5V. For each set of these voltages, dedicated rectifiers, capacitors, and voltage regulators are used. Our design will be similar to this design however for the sake of simplicity our design will not be offering dual (positive and negative) outputs.

Transformer Design

The first major component required for the power supply construction is a power transformer. A power transformer is a magnetic device which converts AC electrical power at one level to AC electrical power at another level. The power transformer is

constructed by winding primary and secondary coils over a laminated magnetic core made from ferromagnetic material. The secondary and primary coils of the transformer are

electrically isolated from each other and all voltages and currents are produced by the virtue of magnetic induction (Faraday's law). The construction of a simple power transformer is shown as following:
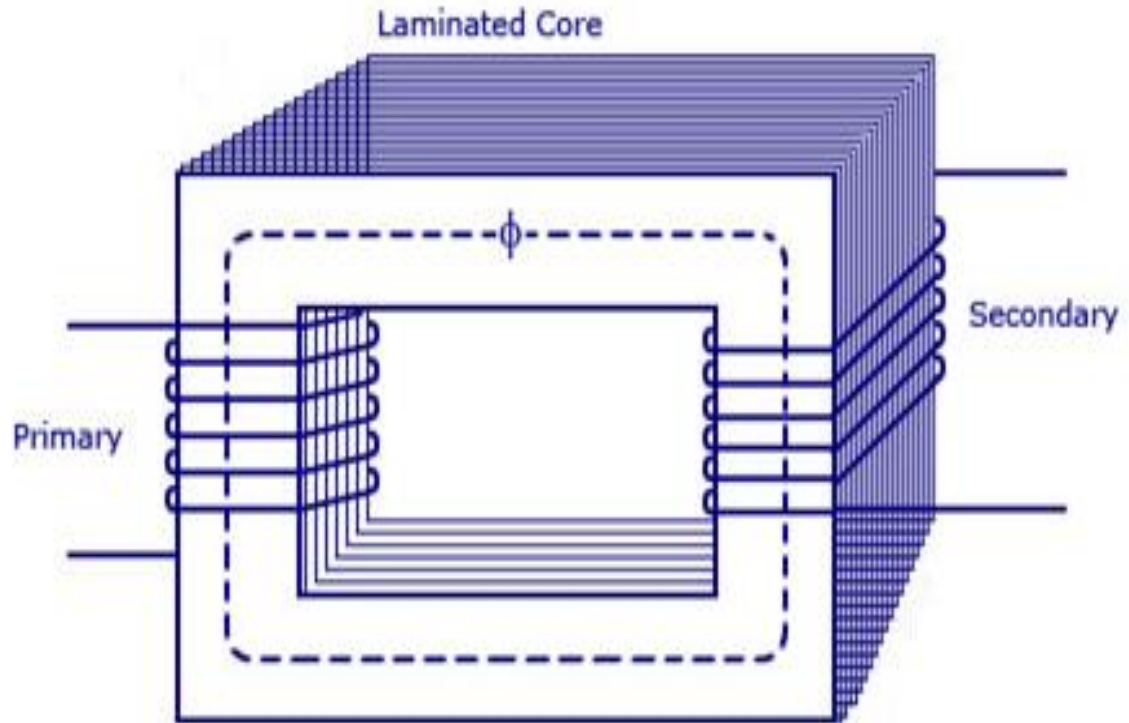
Figure 27, **Construction of a power transformer**

There are two main types of power transformers: step up and step down. In a step-up transformer, the secondary voltage is higher than the primary voltage whereas in step down transformers, the primary voltage is higher than the secondary voltage. For the design of our project power supply, a step-down transformer will be required to supply the low voltage components. This is because we need to convert 120VAC to 5 or 3.3VDC.

In a transformer, the primary and secondary voltages are in direct relation to the number of turns in the primary and secondary coils. The ratio of primary and secondary turns is known as the turns ratio of the transformer. For a step-down transformer the primary coil has a higher number of turns as compared to the secondary coil. The relationship between the turns ratio and the transformer voltages is shown in the following diagram:

Lines of magnetic Flux linking
primary & secondary windings

PRIMARY
N_P turns

SECONDARY
N_S turns

AC Supply

Iron core

Load

$$\frac{\text{The number of primary turns } N_P}{\text{The number of secondary turns } N_S} = \frac{\text{The primary voltage } V_P}{\text{The secondary voltage } V_S}$$

$$\frac{\text{The number of secondary turns } N_S}{\text{The number of primary turns } N_P} = \frac{\text{The primary current } I_P}{\text{The secondary current } I_S}$$
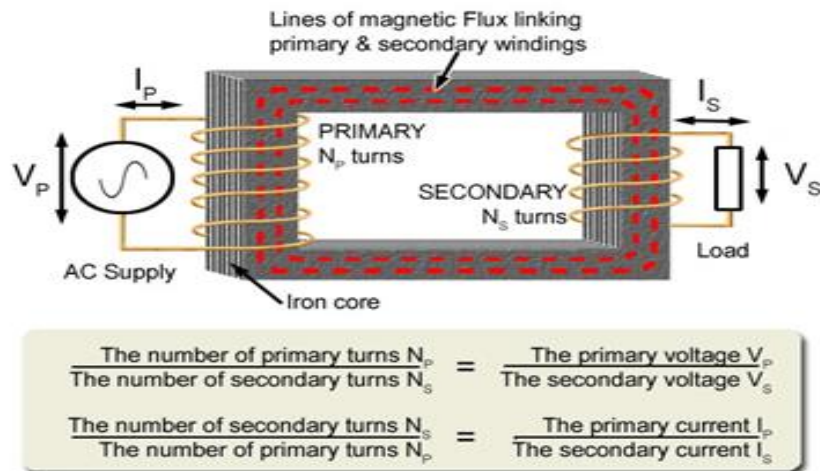
Figure 28, **Transformer turns ratio**

For the preliminary design, we assume that the maximum output current of our power supply is 2A. Hence, the output power is 10W for 5VDC and 6W for 3.3VDC output ports.

Once the transformer converts high voltage AC to low voltage AC, it will be converted to DC using a rectifier and a filter capacitor. Some losses and voltage drops will occur at these stages. We assume a voltage loss of 2V across diodes will occur. Therefore, taking into account an approximate voltage loss of 2V, we will need 8VAC at the secondary of the transformer. Subtracting 2V losses from the 7.2VDC, we get 5.2V which is the required DC voltage. Therefore, the turn ratio of the transformer for 5VDC circuit is 17. Similarly, we can calculate the turn ratio of the transformer for the 3.3VDC circuit. If the secondary AC voltage is 6V, then the equivalent DC voltage is 4.5V. Subtracting 2V losses from this value we get 3V at the output. Hence, for the 3.3V output circuit, the turn ratio of the transformer is 24.

We could have used the same transformer for both the circuits (5V and 3.3V) however in that case the power waste would have been higher, and the power conversion efficiency of the power supply would have decreased. Using two separate transformers for both circuits, the conversion efficiency is improved however the main disadvantage of this scheme is the increase in cost. The transformer is usually the most expensive component in a power supply and using two separate transformers would impact the cost significantly.

We are now going to use a simulation program to verify the primary and secondary voltages of the transformers. We are going to use MULTISIM simulation software in order to simulate the power supply circuit. In this section, only the transformer voltages will be determined using the transformer.
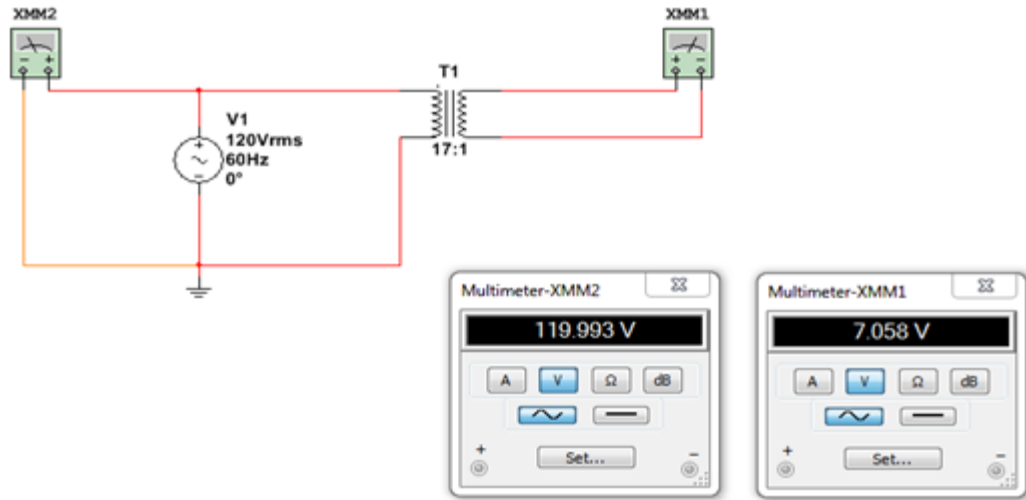
Figure 29, **Transformer design for 5VDC output circuit**

In the above simulation schematic, the input voltage of the transformer is 120VAC RMS. The turn ratio of the transformer is set to 17:1. Hence, an output voltage of 7V is obtained at the secondary of the transformer. Subtracting 1.7V for DC side losses, we will get an output of 5VDC. Now we will repeat the same design procedure for the 3.3VDC output circuit. The transformer simulation for 3.3VDC circuit is as following:
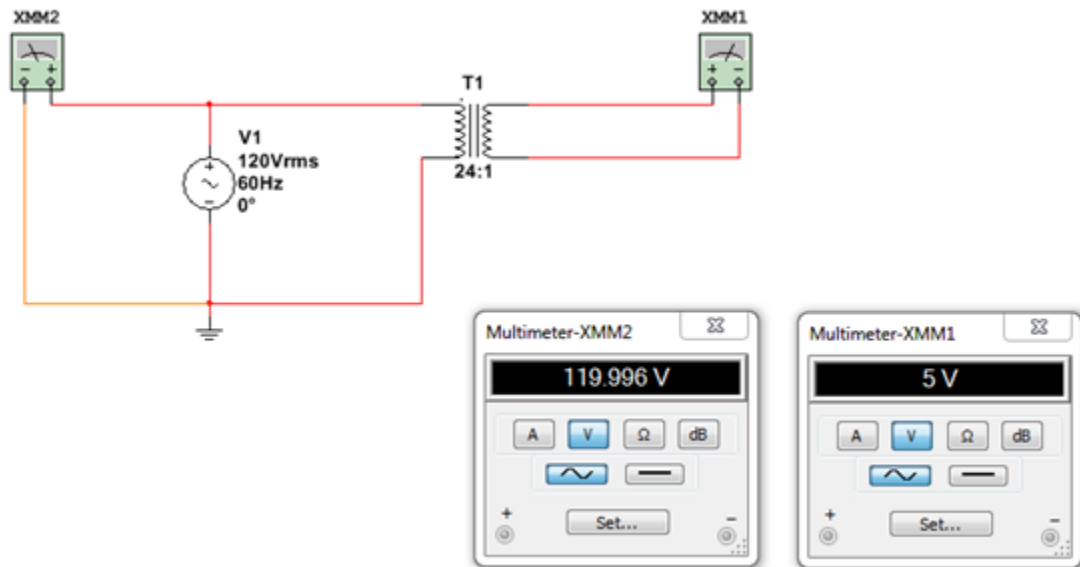
Figure 30, **Transformer design for 3.3VDC output circuit**

The turn ratio for the 3.3VDC circuit is set to 24V so that an output AC voltage of 5V is obtained at the secondary of the transformer. Subtracting 1.7V losses from 5V we get

3.3VDC output. Through these simulations, the correctness of the transformer calculations is verified and therefore we can proceed to the next step in power supply design.

Over-current Protection (Fuse)

The nominal RMS voltage in North America is 120VAC however due to faults in power systems, this voltage can fluctuate significantly. The fault currents and voltages can easily damage sensitive electronic devices. Hence, it is important to provide safety features in the design of power supply. Currents higher than the rated currents of the components can easily damage components like transformers, capacitors, and diodes. Therefore, a fuse is included at the primary side of the transformer for over current protection. A fuse is a thin metallic wire rated at a particular current value. As soon as the input current goes beyond the rated current of the fuse, it melts down and breaks the circuit and protects the circuit's components.

Figure 31, **2A fuse device for over current protection**

The fuse is included in the following way in our power supply circuit.
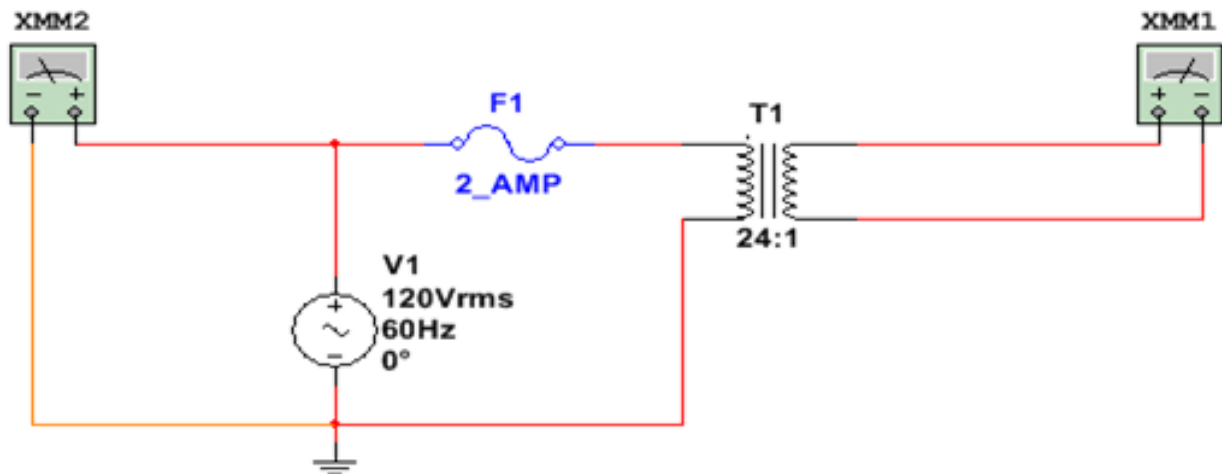
Figure 32, **Fuse in series with the transformer**

As shown in the schematic, the fuse is placed in series with the circuit which has to be protected. As the fault current passes through the fuse, it blows up and interrupts the current flow. Thus, the fault current is hindered from flowing through the sensitive circuit and overcurrent protection is achieved.

Over-Voltage Protection (Varistor)

In the previous section, the operation of the fuse is explained in context of over current protection. However, current protection alone is not sufficient. Transient voltage surges and voltage swells can also damage the components. Therefore, it is also important to include over-voltage protection in the power supply circuit. There are many different ways and techniques of implementing over-voltage protection. One of the simplest and easiest ways is to place a varistor in parallel with the circuit which is to be protected.

A varistor is an electronic component whose resistance varies with applied voltage. The characteristics of the varistor are similar to those of a diode. At normal voltages, the resistance of the varistor is quite high, therefore very little current flows through it. However, as the magnitude of the voltage increases, the resistance of the varistor decreases. This way, most of the fault current flows through the varistor and the primary circuit is saved. Varistor is placed in series with the circuit which is to be protected.

Figure 33, **Metal oxide varistor**
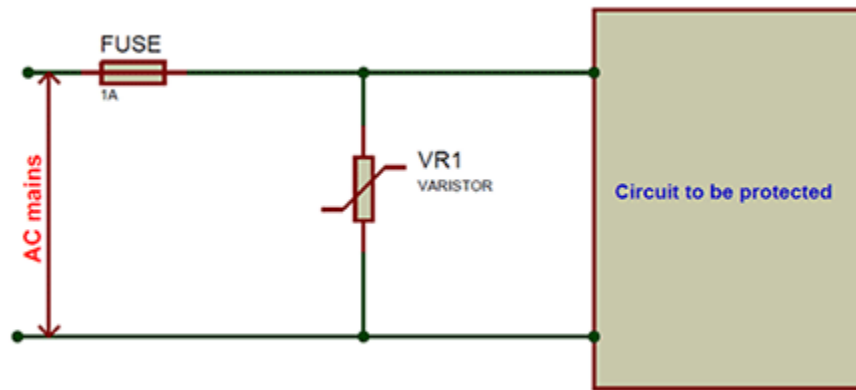
Figure 34, **Placement of fuse and varistor in the circuit**

Using a combination of fuse and varistor at the input side of the power supply, the hazard of over current and over voltage can be reduced. These devices protect our circuit against high voltage transients and fault currents which can cause damage to the electronic components. These devices provide an efficient and cost-effective way of protecting the electronics circuits.

Rectifier

Any electronic circuit which converts AC signal to DC signal is referred to as a rectifier. There are many different types of rectifiers such as half wave, center tapped, and full wave rectifiers. In addition, rectifiers can be constructed either using diodes or thyristors. A diode is an uncontrolled device and does not have any control terminal. A thyristor on the other hand is a controlled device, this means that the powering up of the diode can be controlled by providing a pulse to its gate terminal. The symbols of diode and thyristor are shown as following:
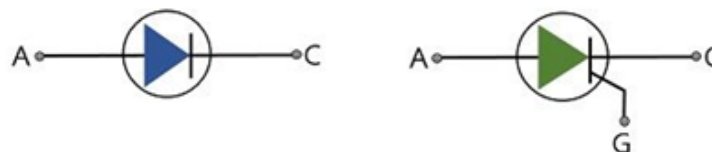
Figure 35, **Diode and thyristor symbol**

Silicon Controlled Rectifier

As mentioned earlier, both thyristors and diodes can be used for constructing rectifiers. A thyristor-based rectifier is often known as SCR or silicon-controlled rectifier. The angle of the thyristor can be controlled using a gate pulse which determines the average output voltage. The angle of the SCR is shown in the following figure:
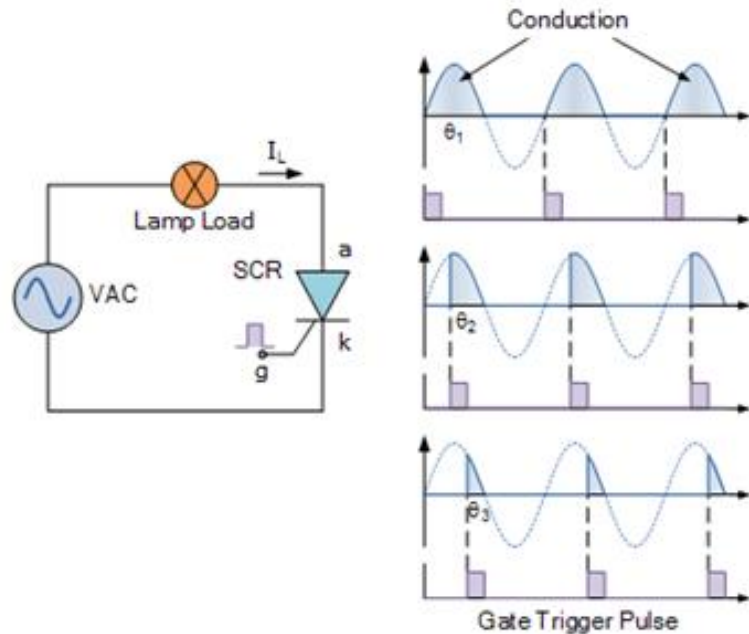
Figure 36, **Angle of SCR**

The control circuitry required for the SCR makes it more complex as compared to the diode rectifiers. Therefore, most of the common power supplies use diodes instead of SCRs. By using diodes, the complexity and cost of the rectifier is reduced.

Diode Rectifiers

We have established the fact that diode rectifiers are simpler and cheaper in construction therefore we will be using diodes for the construction of rectifiers for our power supply. However, there are many different types of rectifiers and we need to choose a suitable rectifier type for our power supply. The three main types of single-phase diode rectifiers are:
1. Half wave rectifier
2. Center tapped rectifier
3. Full bridge rectifier

Half Wave Rectifier

The half wave rectifier configuration makes use of a single diode. Due to this fact, this solution is very cost effective, however the performance of this circuit is not very satisfactory. The circuit for the half wave rectifier is shown as follows:
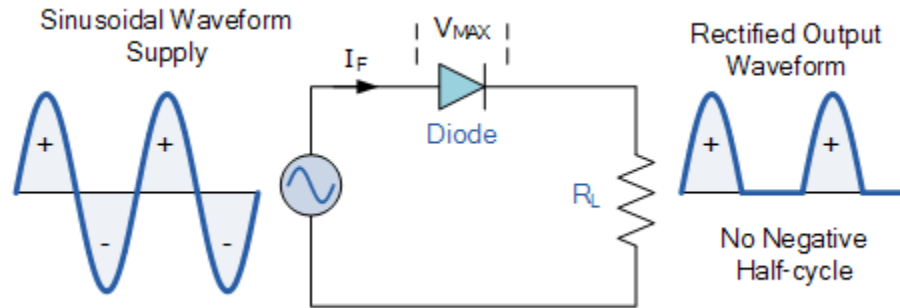
Figure 37, **Half wave diode rectifier**

The diode is a unidirectional device which allows current in only one direction. Hence in the above circuit, the diode allows current flow only during the positive half cycle of the input voltage. During the positive half cycle, the anode voltage is higher than cathode voltage and thus the diode conducts current. During the negative half cycle of the input voltage, the diode is turned off and does not conduct the current. From the output signal of the rectifier we can see that there are no negative half cycles present. This output signal is a DC signal because it is unidirectional. However, this DC signal is not smooth and contains a large amount of ripple content. Due to this large ripple content, the power quality of half wave rectifiers is not so efficient. The DC component of the output signal of the half wave rectifier is 0.45VDC.

	Full-Wave Rectification	Half-Wave Rectification
Circuit Configuration		
Input Voltage Waveform		
Voltage Waveform After Rectification		
Voltage Waveform After Rectification Smoothing		

Figure 38, **Full-wave and half-wave rectification**

This comparison chart above tells us that more than 50% of the input power is lost in a half wave rectifier circuit. This can also be confirmed in an intuitive way as all the negative half cycles are wasted and do not contribute to the output signal. Thus, the half wave rectifier circuit is very inefficient and is not suitable for our power supply design. The output signal of the half wave rectifier has large ripple content. This ripple content can be reduced to some degree using a large filter capacitor. However, there are constraints on the size and cost of the filter capacitor. After placing a filter capacitor of 500uF in parallel with the load, following output voltage we notice that there is a significant improvement in the output

waveform of the half wave rectifier after placing the filter capacitor. However, the fact remains the same that a large amount of input power is wasted in half wave rectifiers which decreases conversion efficiency. Due to this reason, we would not consider a half wave rectifier for the design of our power supply system.

Center Tapped Full Wave Rectifier

Contrary to half wave rectifiers, the full wave rectifiers make use of both the cycles of the input voltage and therefore their conversion efficiency and transformer utilization factor is significantly higher than the half wave rectifier. There are two common types of full wave rectifiers: center tapped and bridge. We will first discuss the center tapped configuration and will determine its suitability for our power supply design. The schematic diagram of the center tapped rectifier is shown as following:
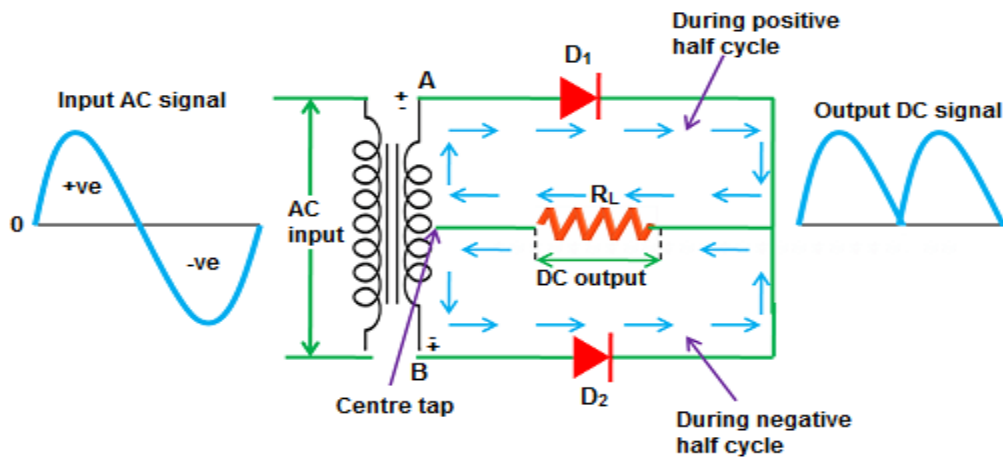
Figure 39, **Center tapped full wave rectifier**

The center tapped rectifier requires a center tapped transformer. This configuration uses two diodes and therefore is more cost effective as compared to the full bridge rectifier which makes use of four diodes. However, the transformer required for this rectifier is heavier and more expensive. The transformer utilization factor of this rectifier is also less as compared to the full bridge rectifier. The diodes D1 and D2 conduct alternatively during the positive and negative half cycles. During the positive half cycle, diode D1 conducts whereas during the negative half cycle diode D2 conducts. As a result of this, both the half cycles are utilized and conversion efficiency is improved. The ripple factor of this rectifier is also better than that of a half wave rectifier and due to this reason the value of the filter capacitor is smaller. One shortcoming of this configuration is that the PIV of the diodes must be at least twice the value of secondary voltage. We can see from the below comparison that the center tapped and full-wave rectifiers have similar characteristics and can be equally efficient. However, the added cost of the center tapped transformer is unnecessary since we can achieve the same characteristics and efficiency using the full bridge rectifier and save the cost of the transformer. Due to this limitation, this rectifier is not suitable for our applications.

Parameters	Center tapped full wave rectifier	Bridge rectifier
Number of diodes	2	4
Maximum efficiency	81.2%	81.2%
Peak inverse voltage	$2V_m$	V_m
Vdc(no load)	$2V_m/\pi$	$2V_m/\pi$
Transformer utilization factor	0.693	0.812
Ripple factor	0.48	0.48
Form factor	1.11	1.11
Peak factor	$\sqrt{2}$	$\sqrt{2}$
Average current	$I_{dc}/2$	$I_{dc}/2$
Output frequency	2f	2f

Table 4, **center tapped full wave rectifier and bridge rectifier**

Full Wave Bridge Rectifier (5VDC)

The final type of rectifier that we are going to analyze is the full bridge diode rectifier. This diode rectifier uses four diodes and has the best characteristics as compared to the previous two configurations. Full bridge rectifier is the most widely type of rectifier in power supply

circuits. This rectifier is available in IC packages and can also be constructed using four diodes. The schematic diagram of this rectifier is provided as follow:
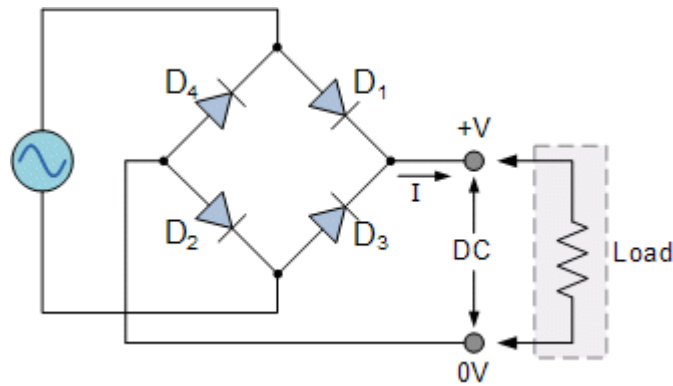
Figure 40, **Full bridge rectifier**

From the schematic we can see that the circuit consists of four diodes. During the positive half cycle, the diodes D1 and D2 conduct whereas during the negative half cycle, diode D3 and D4 conduct. In this way, both the half cycles of input voltage are utilized and no power is wasted. Hence the conversion efficiency of this configuration is good. In addition, this type of rectifier does not need any special type of transformer and thus cost is saved. The ripple content of this rectifier is also less as compared to the other two configurations. Nonetheless, ripple is present in the output waveform and a filter capacitor is required. Due to the simultaneous conduction of two diodes, the voltage drop in this rectifier is double compared to the other configurations. If the voltage drop across a single diode is 0.7V, then the total voltage drop in the rectifier is equal to 1.4V. The frequency of the output voltage is double than the input signal frequency and the magnitude is expressed as following:

The input and output signal waveforms of the full bridge rectifier are presented in the following:
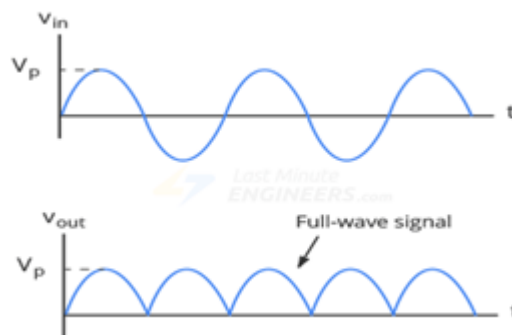
Figure 41, **Input and output waveforms of the full bridge rectifier**

41

The circuit schematic created in the MULTISIM simulation software is presented as following:
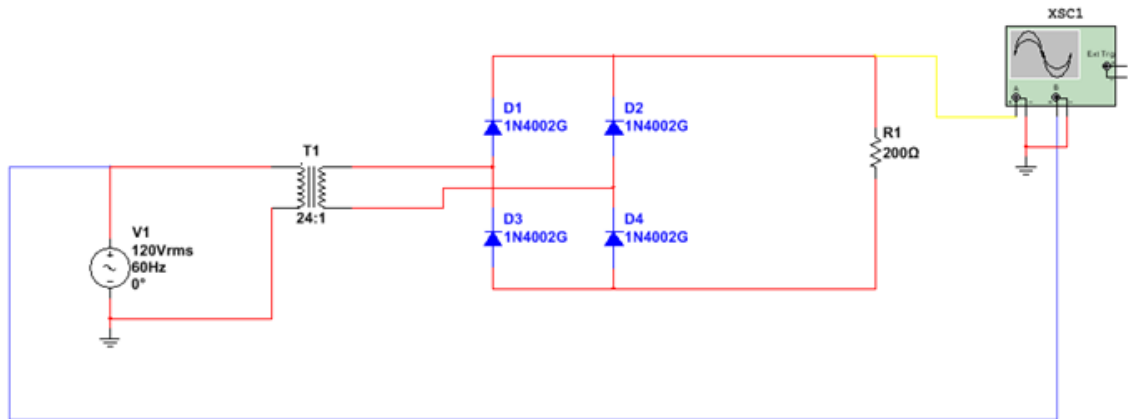
Figure 42, **MULTISIM schematic of full bridge diode rectifier**

For an input voltage of 120VAC, the output voltage provided by this rectifier is 5VDC. The voltage at this level is obtained when the transformer turn ratio is 24:1 and load resistance is 200Ohm.
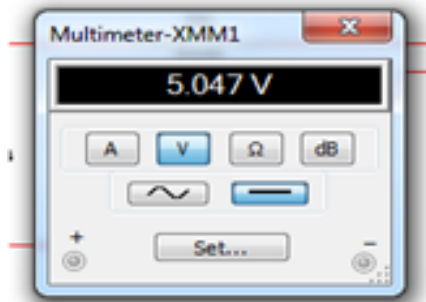
Figure 43, **Output voltage of rectifier for turn ratio of 17:1**

The input and output waveforms of the full bridge diode rectifier are provided in the following figure.
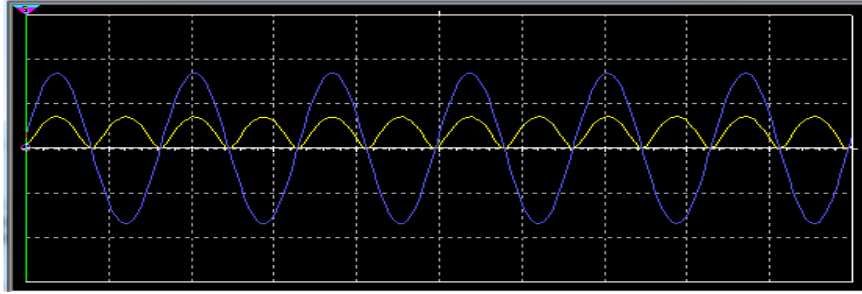
Figure 44, **Input and output waveforms of the full bridge diode rectifier**

From this figure we can see that both the half cycles of the input voltage are reflected in the output. Therefore no power is wasted by this rectifier. However, there are still two main problems with this rectifier.

1. Ripple
2. Voltage drop

The problem of voltage drop across the diodes is something we have to compromise with. Each diode causes a voltage drop of around 0.7V. Hence the total voltage drop across the rectifier is 1.4V. This voltage drop must be taken into account when designing the power supply.

The second problem is that of ripple. The performance of this rectifier is better than half wave and center tapped rectifiers however the ripple still exists. In order to reduce the ripple, a filter capacitor needs to be used in parallel to the load. The selection of capacitor value is often done through trial and error. Therefore a few trails will be conducted in simulation to select an appropriate capacitor value which provides adequate performance at low cost. First of all, we select a capacitor of 3500uF and check its effect on the output voltage waveform.
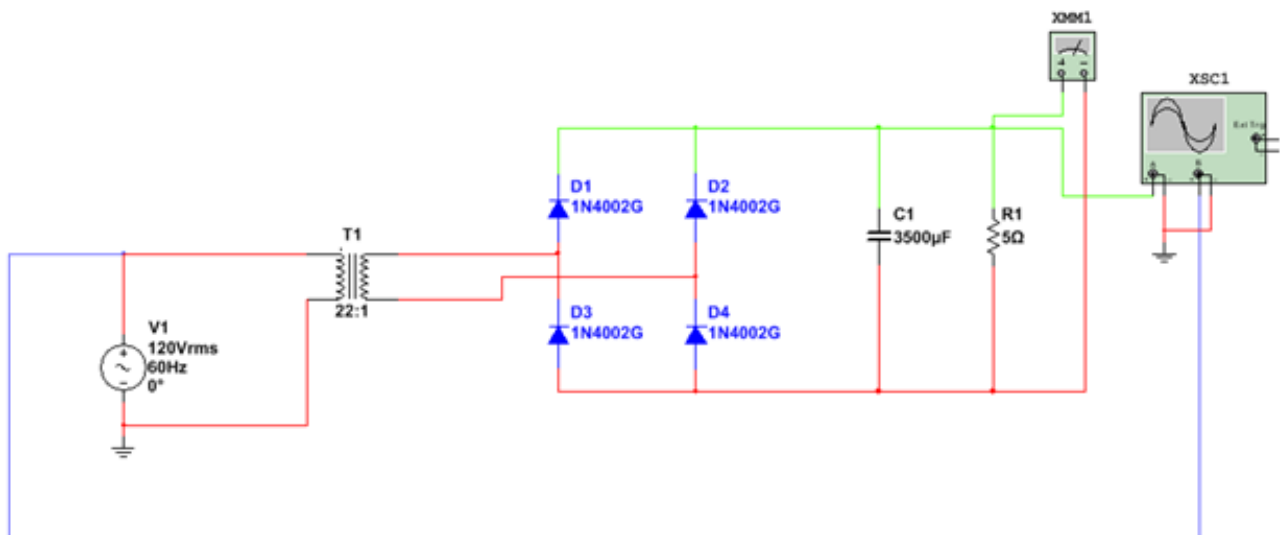
Figure 45, **Full wave diode bridge rectifier with 3500uF filter capacitor**

We notice that the value of the load resistor is changed to 5Ω. This value is selected because the required output current of the power supply is 1A. When the output voltage is 5V and load resistance is 5Ω then an output current of 1A is obtained. Hence, the output power of our designed power supply is 5W. The input and output voltage waveforms with 3500uF filter capacitor are provided as follow:
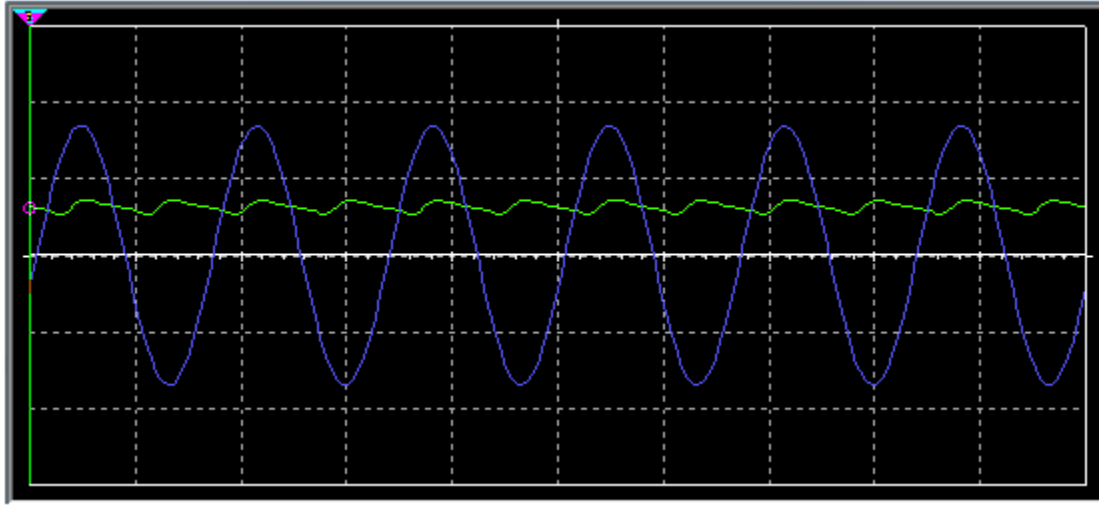
Figure 46, **Input and output voltage waveforms with 3500uF capacitor**

The ripple has significantly reduced and the power quality is enhanced considerably. The output voltage waveform is more close to DC than the previous waveform. The output voltage of the rectifier in this case is as follow:
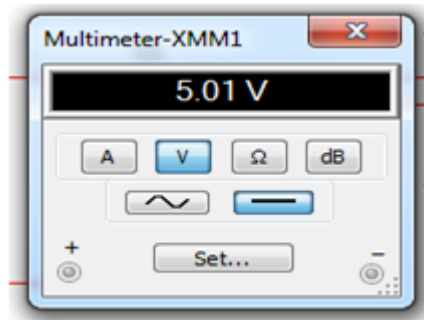
Figure 47, **Output voltage of the rectifier with 22:1 turns ratio and 3500uF capacitor**

Since the required output voltage of 5VDC is obtained with an output current of 1A, we can say that the design objectives have been met. Hence, the filter capacitor of 3500uF will be used for our power supply circuit of 5VDC.

Full Wave Bridge Rectifier (3.3VDC)

In the previous section, we have designed a full wave diode bridge rectifier circuit with an output voltage of 5VDC. However, according to the design requirements, the power supply must provide the provision for additional 3.3VDC. Therefore a separate rectifier circuit needs to be built for the 3.3VDC output. The design of this circuit is discussed in this section. For this subsection of the circuit, an output voltage of 3.3VDC is required whereas the maximum output current is 1A. Therefore, a load resistor of 3.3Ω is needed to obtain an output current of 1A. The circuit schematic is shown as following:
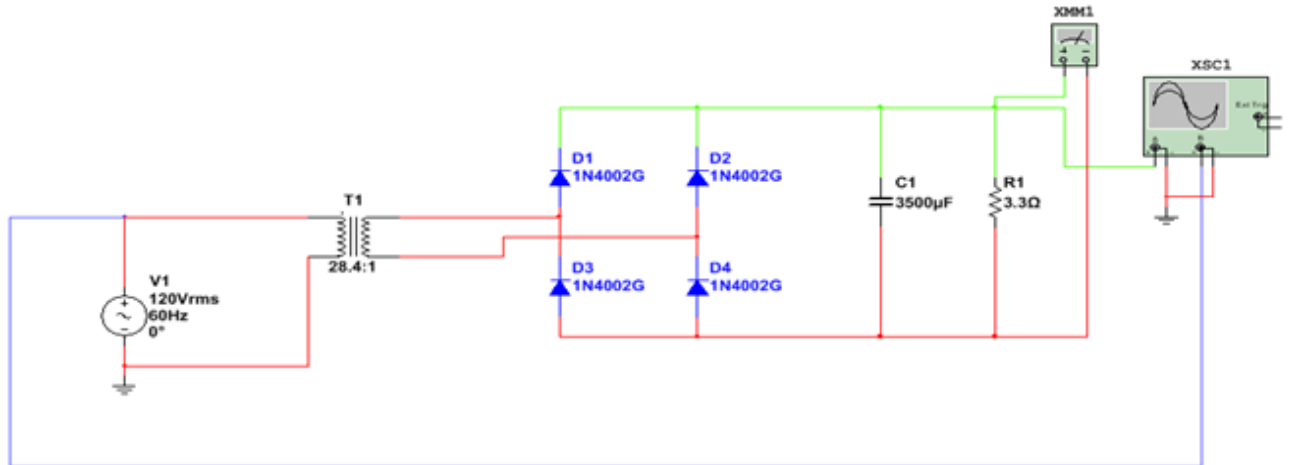
Figure 48, **Power supply circuit with 3.3VDC / 1A output**

We notice that the turn ratio of the transformer is set to 28.4 whereas the value of the filter capacitor is kept the same. For this circuit, the input and output voltage waveforms are shown as following:
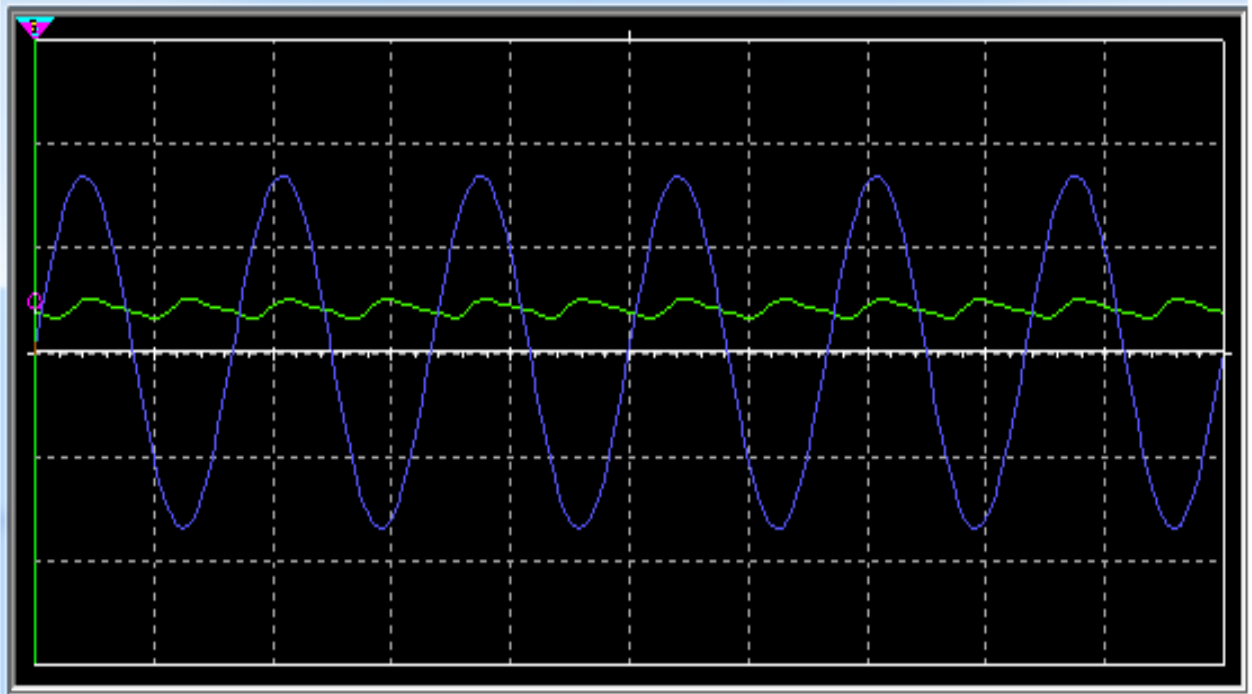
Figure 49, **Input and output voltage waveforms of the 3.3VDC circuit**

From these waveforms we can see that the ripple voltage is at an acceptable level. Also the output voltage of the circuit is as following:
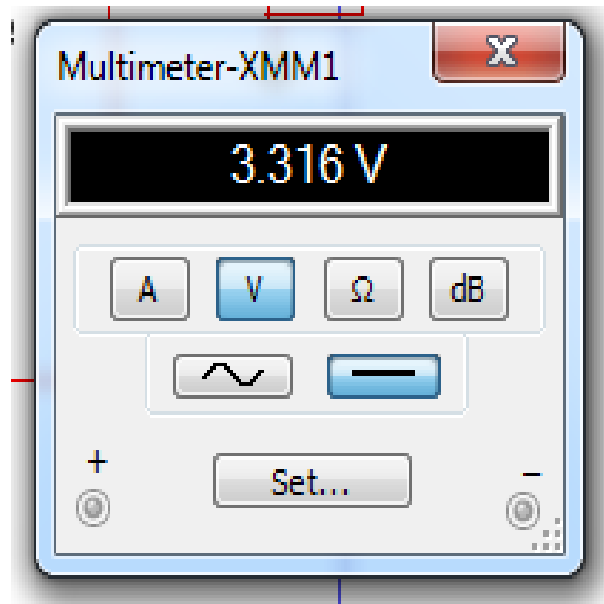
Figure 50, **Output voltage of the 3.3VDC circuit**

The output voltage provided by this circuit is 3.3VDC at 1A which is according to the design requirements. Hence we can say that this design fulfills the design requirements.

Voltage Regulator

In the previous section we observed that the rectifier output consisted of large ripple content and therefore it was not a pure DC signal. In order to smoothen the ripple, we introduced a large capacitor of 3500uF in the circuit which significantly reduced the ripple content. However, from the analysis of waveforms we can see that even after filtration, the output voltage signal is not perfectly smooth and still consists of some ripple content.

The solution to remove the residual ripple from the output voltage is to use a voltage regulator. Using a voltage regulator, we can obtain a stable and smooth DC signal at the output. A voltage regulator is basically an active feedback electronic circuit which takes in a higher voltage and provides a stable lower voltage at its output. There are two main types of voltage regulators: linear and switching. Linear voltage regulators use op amps and feedback loops to maintain a stable voltage level. Switching regulators consist of high frequency DC-DC choppers which provide the regulated output voltage. Linear voltage regulators are more widely used due to their simplicity and low cost. For 5VDC output, the most common voltage regulator IC is LM7805. The LM78XX family of regulators comes with a heat sink. The reason for using a heat sink is that the voltage regulator dissipates extra voltage in the form of heat and therefore the IC package gets heated. Without a proper heat sink, the voltage regulator will get burnt.
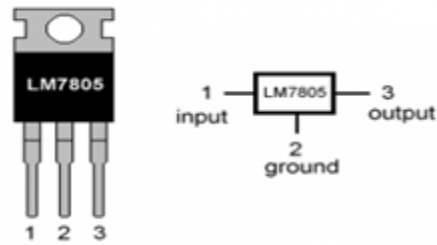
Figure 51, **LM7805 voltage regulator**
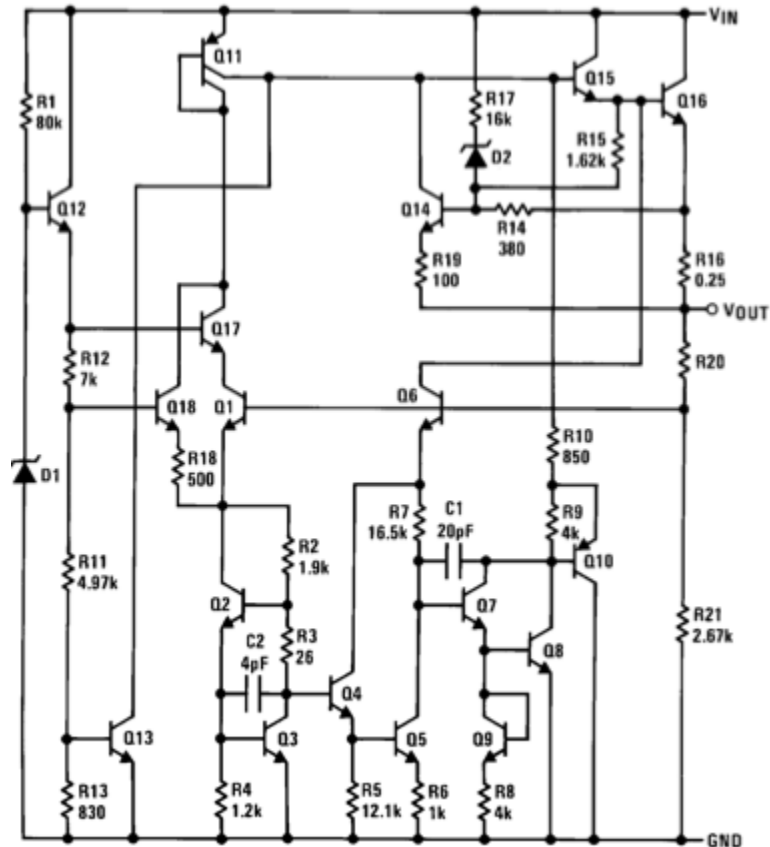
Figure 52, **functional block diagram of LM7805**

Now we will place the 7805 voltage regulator in our 5VDC sub-circuit in order to get a stable 5VDC signal across the load resistance. The MULTISIM simulation circuit with the voltage regulator is provided as follow:
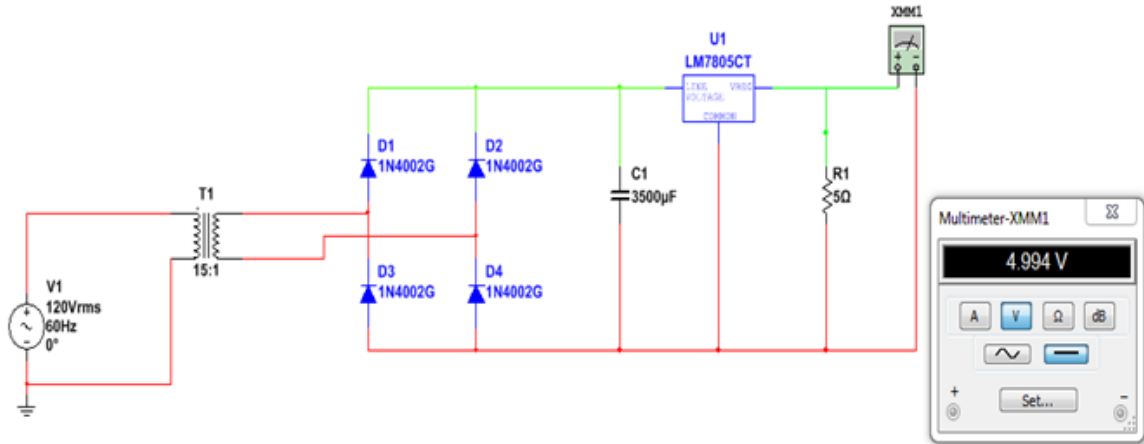
Figure 53, **5VDC power supply circuit with voltage regulator**

From the above figure, we can see that an exact voltage of 5VDC is obtained at the output of this circuit. The input and output waveforms of this circuit are as following:
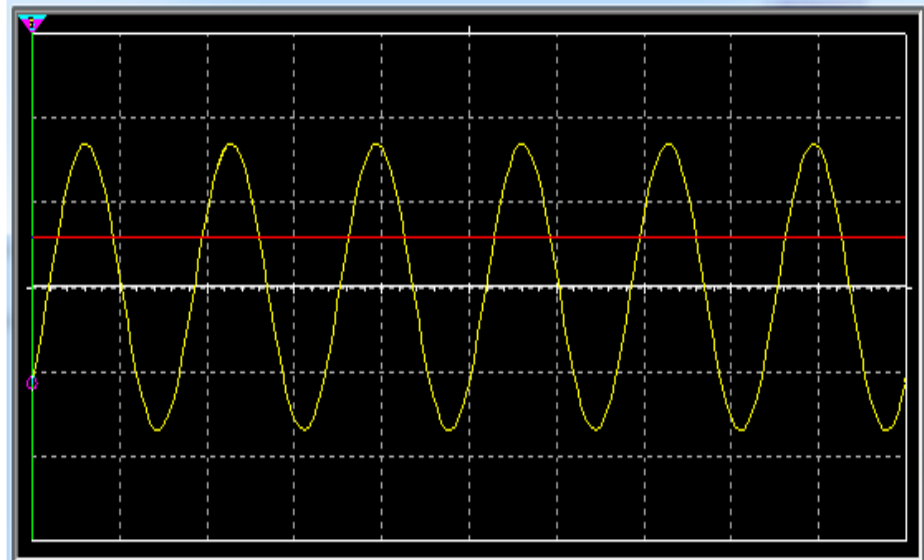
Figure 54, **Input and output waveforms of the 5VDC rectifier with voltage regulator**

From the circuit we can see that the output signal (red) is a straight and smooth DC signal without any irregularities or ripples. Hence, by using a voltage regulator at the output of the circuit we obtain a ripple free output voltage signal. For the 3.3VDC sub-circuit, the same design will apply. The only difference is that instead of using a 5V regulator, we will use a 3.3V regulator which will provide a stable 3.3VDC signal at the circuit output. We were able to achieve our goal when it comes to voltage and current outputs using a full bridge rectifier. We were also able to get rid of the ripple which is a huge concern to the circuitry of our project. However, the use of the linear voltage regulator LM7805 is not cost effective. Although the regulator as a part is fairly cheap, the voltage input requirement

cannot exceed 35v as can be seen in the datasheet below. Since the LM7805 and most of the typical regulators are not designed to operate with the standard 120v, the addition of a step-down transformer is inevitable. After researching multiple linear voltage regulators from different manufacturers such as the Onsemi NCV51460 and the Microchip Technology MAX15006 and MAX15007, we can conclude that most of these popular voltage regulators require a lower voltage input. This fact is an inconvenience since we will need to add a heavier transformer to step down the outlet voltage to a lower voltage that would be suitable for these regulators. Due to these limitations, we will continue investigating other options to design a cost effective power supply for our project.

		MIN	MAX	UNIT
DC input voltage			35	V
Internal power dissipation[3]		Internally Limited		
Maximum junction temperature			150	°C
Lead temperature (soldering, 10 sec.)	TO-3 package (NDS)		300	°C
	Lead temperature 1,6 mm (1/16 in) from case for 10 s		230	°C
Storage temperature		−65	150	°C

Figure 55, **Absolute Maximum Ratings for LM7805**

The MAX15006/MAX15007 is another option that we have investigated. Although these voltage regulators have the advantage of providing a fixed 5v output,the maximum output current is +/_ 50mA. This rating is not sufficient for the operation of the microcontroller, therefore we will not consider this regulator for our project.

Figure 56, **Absolute Maximum Ratings MAX15006/MAX15007**

```
IN to GND..............................................................-0.3V to +45V
EN to GND ............................................................-0.3V to +45V
OUT, FB to GND ...................................................-0.3V to +12V
OUT Short-Circuit Duration .......................................Continuous
Maximum Current Into Any Pin (except IN and OUT)......±50mA
Continuous Power Dissipation (T_A = +70°C)
    TDFN (derate 23.8mW/°C above +70°C)..................1904mW
    SO (derate 18.90mW/°C above +70°C)...............1509.40mW
```

The NCV51460 provides high performance as well as low power precision voltage reference and accuracy. This regulator has low power dissipation but can supply an output current up to 20 mA at a 3.3 V fixed output voltage. It also provides efficient line and load

regulation characteristics, and it is designed to be stable with or without an output capacitor. This regulator includes an internal Short Circuit and Reverse Input Voltage Protection. Despite all these qualities, the NCV51460 is not suitable for our project due its low Quiescent Current as can be seen on the data sheet.

Parameter	Test Conditions	Symbol	Min	Typ	Max	Unit
Output Voltage		V_{OUT}	3.267 (−1%)	3.3	3.333 (+1%)	V
Line Regulation	$V_{IN} = V_{OUT} + 0.9$ V to $V_{OUT} + 2.5$ V $V_{IN} = V_{OUT} + 2.5$ V to $V_{OUT} + 20$ V	Reg_{LINE}	− −	150 65	500 130	ppm/V
Load Regulation	$I_{OUT} = 0$ to 100 μA $I_{OUT} = 0$ to 10 mA $I_{OUT} = 0$ to 20 mA	Reg_{LOAD}	− − −	1100 150 120	4000 400 400	ppm/mA
Dropout Voltage	Measured at $V_{OUT} - 2\%$ $I_{OUT} = 0$ mA $I_{OUT} = 10$ mA	V_{DO}	 − −	 0.65 0.9	 0.9 1.4	V
Quiescent Current	$I_{OUT} = 0$ mA, $T_A = 25°C$ $I_{OUT} = 0$ mA, $0°C \le T_A \le 100°C$	I_Q	− −	140	200 220	μA
Output Short Circuit Current	$V_{OUT} = 0$ V, $T_A = 25°C$	I_{SC}	−	80	−	mA
Reverse Leakage	$V_{IN} = -15$ V, $T_A = 25°C$	I_{LEAK}	−	0.1	10	μA
Output Noise Voltage (Note 6)	$f = 0.1$ Hz to 10 Hz $f = 10$ Hz to 1 kHz	V_N	− 	12 18	− 	$μV_{PP}$ $μV_{rms}$
Output Voltage Temperature Coefficient	$0°C \le T_A \le 100°C$ $-40°C \le T_A \le 125°C$	T_{CO}	− −	18 34	− −	ppm/°C

Figure 57, **NCV51460 ELECTRICAL CHARACTERISTICS**

Linear voltage regulators

Linear voltage regulators use a transistor design to convert an input voltage into a constant specific output voltage. They can be purchased as small integrated circuits which can be easily implemented on any printed circuit board. Linear regulators can provide both positive and negative output voltages. They are available in fixed voltage ratings and as adjustable voltage regulators. Linear voltage regulators provide output voltages from 1 to 40 V with current Load from 1 to 1.5A. The operating voltage requirements for Linear voltage regulators have minimum and maximum ratings and limitations. The minimum operating voltage requirement is determined by the drop off voltage as specified in the datasheet of the linear regulator. The drop off voltage is typically between 2 and 3 V and for 5 V regulators. For such a regulator, a minimum input of 7 to 8 volts would be required in order to supply 5 volts output. The maximum input voltage depends on the series of the regulator and typically reaches a 40 V maximum as an input.

Specification	Linear	Switcher
Line Regulation	0.02%–0.05%	0.05%–0.1%
Load Regulation	0.02%–0.1%	0.1%–1.0%
Output Ripple	0.5 mV–2 mV RMS	10 mV–100 mV$_{P\text{-}P}$
Input Voltage Range	±10%	±20%
Efficiency	40%–55%	60%–95%
Power Density	0.5 W/cu. in.	2W–10W/cu. in.
Transient Recovery	50 µs	300 µs
Hold-Up Time	2 ms	34 ms

Figure 58, **Summary of efficiency and power loss**

A linear voltage regulator would provide sufficient power to our project since the specifications required for our design are 5V and 3.3V. the micro-processor and sensor also require minimum currents which can be easily achievable with a linear voltage regulator. The drop out voltage requirement will not be an issue for our design. Therefore, a linear voltage regulator meets the required specifications and could be used for all the voltage regulation needs for our project. The benefit of using a linear voltage regulator is its simple circuitry since a typical voltage regulator has only three pins. There are no additional needed components required to accomplish the specified voltage regulation. However, the simplicity of the design of linear regulators comes with a critical limitation which is inefficiency. This type of voltage regulator is very inefficient since almost half the output power becomes dissipated as excess heat.
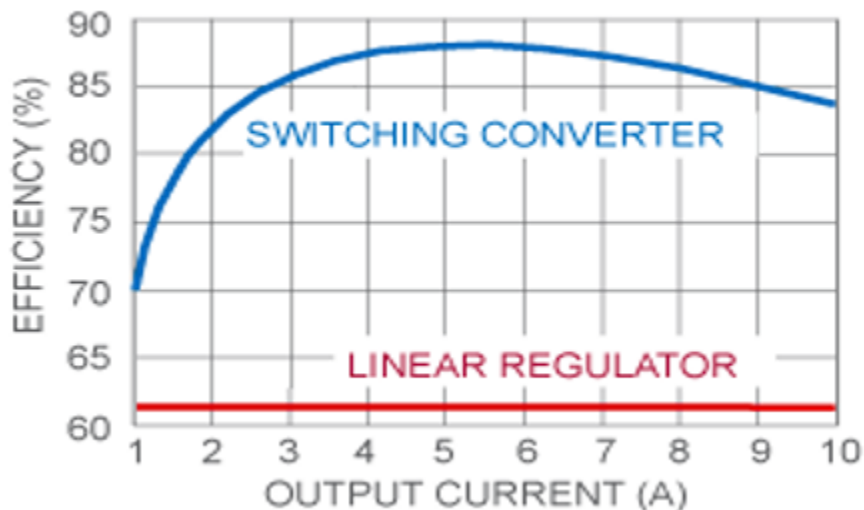
Figure 59, **Efficiency Curve for LDO and Switching Regulators**

Switching voltage regulators

Switching voltage regulators use a combination of transistors which act as switches, they also use inductors and capacitors that are storage devices to provide a constant output voltage. Switching regulators are divided into categories; buck, boost, and buck-boost regulators. A buck converter steps down a higher input voltage to a constant lower output voltage. A boost converter, also known as a step-up converter, provides an output voltage greater than the input voltage. A buck-boost converter provides the same regulation whether the input voltage is greater or less than the output voltage. for the sake of efficiency, a buck voltage regulator will be considered for our project. Just like linear regulators, switching regulators can also be integrated circuits that can operate with up to 40V and 3A.

Switching regulators have the capability to produce an output voltage that is higher than the input, these regulators are more complex and use transistors as switching elements that convert the input voltage into a pulse width modulation PWM. The PWM is converted to DC voltage output using filters, although they create more noise in the circuit, they provide higher efficiency compared to linear regulators. The combination of both regulators is also possible to provide a stable output.
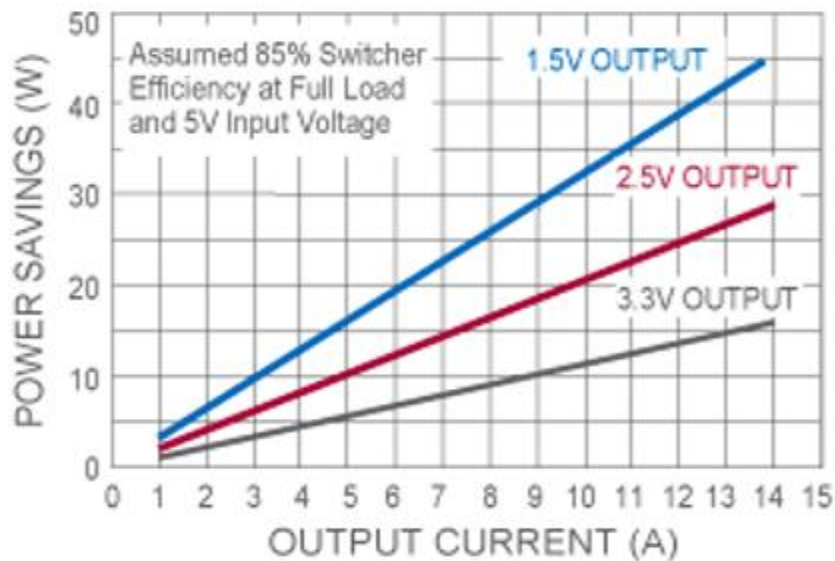
Figure 60, **Power Savings Between LDO and Switching Regulators**

Unlike linear regulators, switching regulators circuitry do not dissipate excess power to heat. This feature makes the efficiency almost double that of the linear regulators. Modern electronic technology uses switching voltage regulators because they are suitable for supplying power to most micro-processors. The main reason why these regulators are used more than linear regulators is because of their efficiency. These regulators can also be purchased as integrated circuits which will make them easily implemented into any circuit design. The shortcoming of switching regulators is that their output voltage has more ripple

than linear regulators, this is mainly due to the inductors embedded in their circuitry. The ripple effect can be compensated by the use of a capacitor.

After investigating these types of regulators, we ended up choosing linear regulators for our application. Despite the fact that switching regulators have higher efficiency, linear voltage regulators provide low ripple output voltage which is critical for the operation of the arduino uno and the wi-fi module.
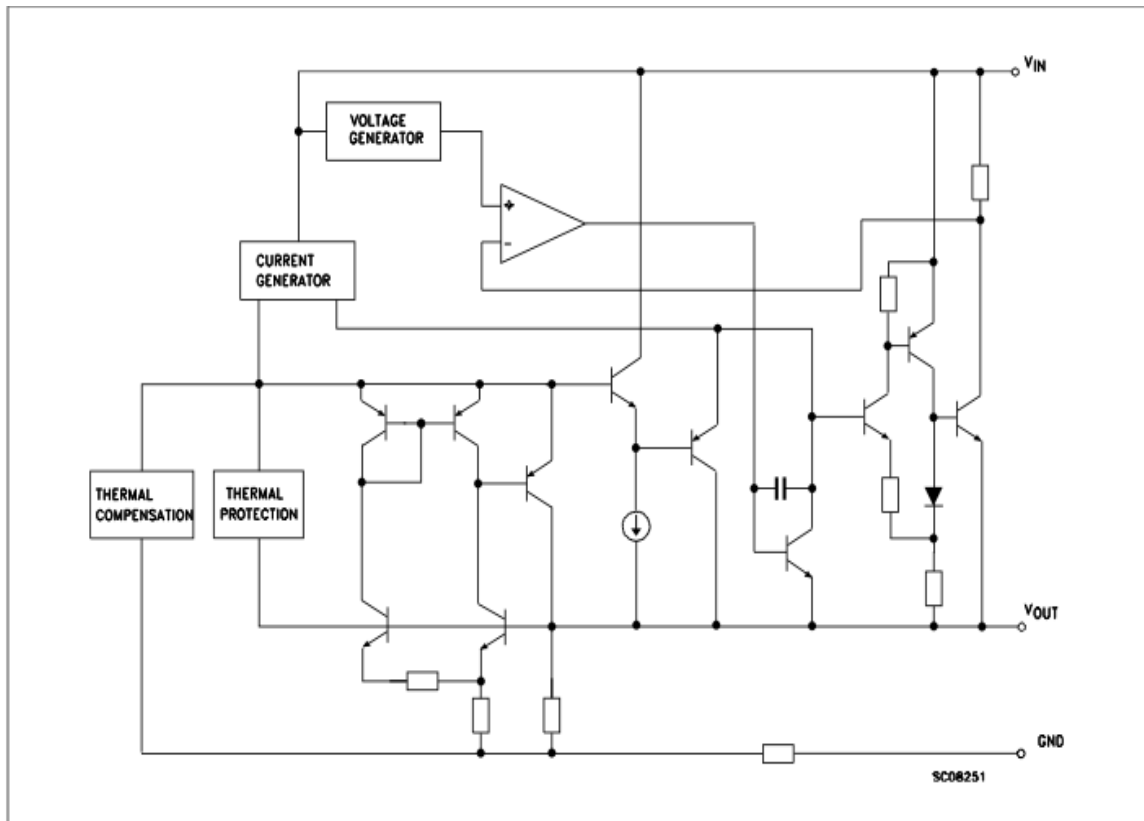
Figure 61, **Block diagram of the LD1117 Low drop fixed output voltage**

Our final choice for the voltage regulator is to use linear regulators that will meet the specifications and requirements of our project. We ended up choosing the Texas instruments LM7805 fixed output voltage to supply the 5v to the main PCB and the ultrasonic sensors. We also ended up selecting the LD1117 to supply the 3.3v required to power the wi-fi module. These fixed output voltage regulators are ideal for our project since they have internal thermal protection and overcurrent protection. These features are desirable for the protection of the circuit since the regulator will open the circuit in case of overcurrent or when the device temperature exceeds the manufacturer's specifications. These linear regulators are rated at 1.5A and 1A respectively which is sufficient to accomplish what we need to power the pet feeder.

Symbol	Parameter	Test conditions	Min.	Typ.	Max.	Unit
V_O	Output voltage	$V_I = 5.3$ V, $I_O = 10$ mA, $T_J = 25$ °C	3.234	3.3	3.366	V
V_O	Output voltage	$I_O = 0$ to 1 A, $V_I = 4.75$ to 10 V	3.168		3.432	V
ΔV_O	Line regulation	$V_I = 4.75$ to 8 V, $I_O = 0$ mA		1	6	mV
ΔV_O	Load regulation	$V_I = 4.75$ V, $I_O = 0$ to 1 A		1	10	mV
ΔV_O	Temperature stability			0.5		%
ΔV_O	Long term stability	1000 hrs, $T_J = 125$ °C		0.3		%
V_I	Operating input voltage	$I_O = 100$ mA			10	V
I_d	Quiescent current	$V_I \leq 10$ V, $I_O = 0$ mA		5	10	mA
I_O	Output current	$V_I - V_O = 5$ V, $T_J = 25$ °C	1000	1200		mA
eN	Output noise voltage	B =10 Hz to 10 kHz, $T_J = 25$ °C		100		µV
SVR	Supply voltage rejection	$I_O = 40$ mA, f = 120 Hz $V_I - V_O = 3$ V, $V_{ripple} = 1$ V_{PP}	60	75		dB
V_D	Dropout voltage	$I_O = 100$ mA		1	1.10	V
		$I_O = 500$ mA		1.05	1.15	
		$I_O = 1$ A		1.15	1.30	
$\Delta V_{O(pwr)}$	Thermal regulation	$T_a = 25$ °C, 30 ms pulse		0.08	0.2	%/W

Figure 62, **Electrical characteristics of LD1117A#33**

3.3 Microcontroller, Sensors and Indicators

Microcontrollers

Three different microcontrollers will be compared: Arduino AG's Uno, Raspberry Pi 4, and Texas Instruments' MSP430 Series. All information is obtained from the respective company's documentation, wikipedia, and open source projects created by other contributors. None of the information below is our work except for basic conclusions that can be drawn from the data. This section will discuss what a microcontroller is, what it should do, and what are some qualities to make a certain microcontroller preferable. Common microcontrollers are compared.

Microcontrollers are the brain of the project and one will be selected to suffice for the project. The microcontroller will need to handle many tasks such as weight measurements, connecting to Wi-Fi, infrared-sensor, and button presses. All of these microcontrollers can handle the tasks required but some do it easier than others. This can be because of libraries, accessibility of parts, or built in functions.

When selecting a microcontroller there are many unique factors that can easily make or break the selection. Power is one quality that can affect the decisions. How much power a microcontroller generally needs for its operation is important. Not only will the microcontroller need power to operate itself but it will also need to draw power for its add-

ons. Wi-Fi integration, sensors, processing, and clocks are such features that may require additional power. All components also have their own manufacturers so their efficiency is not the same. Power also includes how they receive their voltage. Some microcontrollers can run off batteries, some require an outlet, or maybe they have the option to be charged with solar power. A microcontroller that can only work on an outlet may not be picked over one that can work on an outlet or battery. Power failings may result in hazards or inevitable failure of the project.

The components and their availability is crucial not only for the microcontroller but for this Senior Design project. The microcontroller must have the available sensors and other components to meet all the goals required. A good microcontroller not only meets the minimum but must have a variety of choices for a single part that can be replaced so we aren't stuck using a single type of sensor that has little to no support. With many options for components, we can choose one that best suits our needs and is easily affordable for any replacements or replications.

Programming the microcontroller is also a key factor for decision. We as the users will need to make the microcontroller do something, so how we can do it is important. Everything from programming language to IDE to where and how you write the code is important. Some microcontrollers can be directly coded through themselves and others require an external device such as a computer to upload code to be compiled and executed. Libraries and APIs will also heavily influence the decision. If there are many community-made tutorials and code then this can save time and teach other programmers easier, they will be preferred.

The last factor that determines the microcontroller is overall price. A microcontroller could be very cheap but require $100 for a Wi-Fi addon. We need a microcontroller that is priced reasonably well with all of its components being cheap but fairly good quality so if they ever break they can be replaced.

Arduino Uno Microcontroller

Figure 63, **An Arduino Uno board**

56

The first microcontroller is the Arduino Uno. It uses an ATmega328 microchip for its CPU. This 80bit RISC-based microcontroller gives 32KB ISP Flash memory that can read-while-write. It has 32 general registers, serial programmable USART (used to receive and transmit communication with a computer), a programmable timer, and 5 power saving modes. It operates between 1.8 - 5.5 volts and has 32 pins.

One thing that makes the Arduino popular is its ease of use and accessibility. The Arduino is programmed in its own IDE using sketches. The sketches are the files that are used to run programs on the Arduino.

Another thing the Arduino Uno is known for is its open source community libraries and projects. Due to Arduino's massive third party support, many of the add-ons and shields have accompanying libraries made by the manufacturers that allow for easy use. The support that Arduino has allows for easier implementations and less trial and error for code and execution. With time saved from other people's work, more can be done.

Figure 64, **Arduino Uno powered by 9V battery**

The Arduino Uno is priced at around $22, Wi-Fi shield less than $10, and sensors all coming together for around $20 for a total of $60 with any other fees. Many starter kits also are sold for around $50 or $60, that include all of the wires and any additional components such as breadboards or resistors.

The Arduino Uno can be powered with a battery or outlet source, allowing for easier testing and portability. It only operates on 5V DC which can easily be used from a battery and uses around 42mA of current for normal operation. This is around .21 Watts of power.

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13
Length	68.6 mm
Width	53.4 mm
Weight	25 g

Figure 65, **An Arduino Uno datasheet**

Raspberry Pi Microcontroller
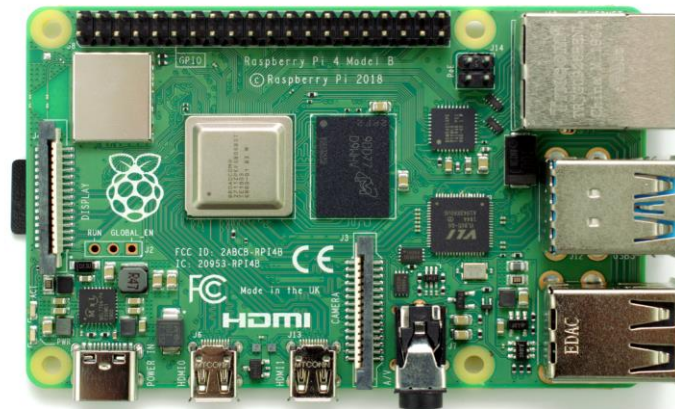
Figure 66, **A Raspberry Pi 4 Model B**

The next microcontroller is the Raspberry Pi 4. The Raspberry Pi 4 is a mini computer loaded with the Linux OS. It can support dual monitors, a mouse, and keyboard; working like a normal computer. The Raspberry Pi uses a 64-bit dual-core processor, allows up to 8GB ram, has a USB 3.0 slot, and built in dual-band 2.4/5GB wireless LAN for internet connection. Storage is held on an external micro-SD card slot. Similar to the Arduino Uno's shields for add-ons, the Raspberry Pi has Pi hats that serve the same purpose.

Due to the Raspberry Pi being a computer rather than a microchip that runs code, there aren't many libraries for it. However, there are still various community projects that can be used for research and information. Another problem from it being a computer is the amount of power used. The Arduino only runs 1 code constantly and has low power modes to turn off settings. The Raspberry Pi is an entire system, using Graphics Cards, CPUs, and peripherals such as mouse and keyboard causing it to drain more power. According to the Raspberry Pi website, it uses 5.1V and minimum 700mA of current for it's basic model, Raspberry Pi Model A. This means at minimum it will use 3.5 Watts of power, almost 16x more than the Arduino Uno. This is however expected with the amount of hardware the Raspberry Pi has built into it.

Using power is not the only problem the Raspberry Pi has, giving it power has limited options. The Arduino Uno could be powered with a battery, outlet, or microUSB/USB-C. The Raspberry Pi can only be powered with a microUSB/USB-C cord, meaning it needs an external USB power bank or another computer.

A Raspberry Pi 4 model B with 2GB ram starts at $35 on many seller sites such as AdaFruit, PiShop, and microcenter. It includes built in WiFi but no displays, fans, heatsinks or sensors. The weight sensor needs to be built on it's own, however there are many tutorials to guide users to build their own sensor. A basic Black and White 16x2 LCD display from Adafruit is around $20, much more pricier than the Arduino Uno.

Processor:	Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz	**SD card support:**	Micro SD card slot for loading operating system and data storage
Memory:	1GB, 2GB, 4GB or 8GB LPDDR4 (depending on model) with on-die ECC	**Input power:**	5V DC via USB-C connector (minimum 3A[1]) 5V DC via GPIO header (minimum 3A[1]) Power over Ethernet (PoE)–enabled (requires separate PoE HAT)
Connectivity:	2.4 GHz and 5.0 GHz IEEE 802.11b/g/n/ac wireless LAN, Bluetooth 5.0, BLE Gigabit Ethernet 2 × USB 3.0 ports 2 × USB 2.0 ports.	**Environment:**	Operating temperature 0–50°C
GPIO:	Standard 40-pin GPIO header (fully backwards-compatible with previous boards)	**Compliance:**	For a full list of local and regional product approvals, please visit https://www.raspberrypi.org/documentation/hardware/raspberrypi/conformity.md
Video & sound:	2 × micro HDMI ports (up to 4Kp60 supported) 2-lane MIPI DSI display port 2-lane MIPI CSI camera port 4-pole stereo audio and composite video port	**Production lifetime:**	The Raspberry Pi 4 Model B will remain in production until at least January 2026.
Multimedia:	H.265 (4Kp60 decode); H.264 (1080p60 decode, 1080p30 encode); OpenGL ES, 3.0 graphics		

Figure 67, **A Raspberry Pi 4 DataSheet**

MSP430 Microcontroller
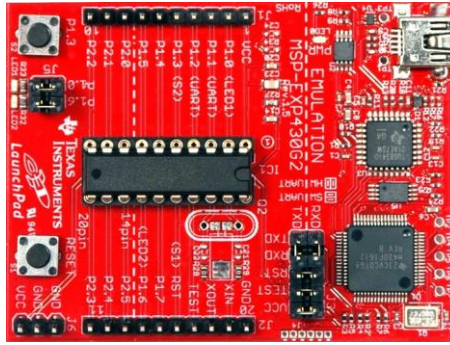
Figure 68, **A MSP-EXP430G2 LaunchPad Board**

The last microcontroller is the MSP430 series. The MSP430 is a 16-bit microcontroller by Texas Instruments that comes in many different forms. According to the TI website, they allow for automation, grid infrastructures, factory automation, and many more. The series have different features with UART, sensors, timers, different pincounts, real time clocks, and advanced sensors. MSP430s are most notable for their power modes, having 7 different low-power modes that turn off different functions, clocks, and sensors to conserve power.

One of the most common MSP430 series microcontrollers is the MSP430FR6989 Launchpad. The FR is priced at $25, very similar to the Arduino Uno. The FR has 2KB RAM, 83 GPIO pins, 2 LEDs: green and red, built in LCD display, UART, and low-power modes.

The FR has 3 clocks, ACLK (Auxiliary Clock), SMCLK (Sub Master Clock), and MCLK (Master Clock), that can run off the 32KHz crystal. The clock timers can all be configured through their respective registers, for example Timer A is controlled with its register: TACTL.

Like other microcontrollers, the FR also has add-ons for it. One of the popular boards to go with the FR is the "Educational BoosterPack MKII", including its own LEDS, joysticks, push buttons, sensors, and LCD display. There is also a wireless network processor that allows the FR to connect to the internet. For example, the CC3100 allows for 2.4GHz wifi connection with 16Mbps throughput. This is a bit pricey at $20 to $30.

The FR is programmable with C language however it can become very complicated due to the heavy reliance on registers. For example, a basic command such as using a switch to turn on the green LED is a hassle due to the amount of register manipulation. The programmer has to manage the flag registers, the clock registers, the LED registers, and the switch registers. Typically all the registers and their bit values are stored in the datasheet, but rather than running a simple function to turn on the LED, a ton of manipulation has to be done.

Non-volatile memory (kB)	128
RAM (KB)	2
ADC	12-bit SAR
GPIO pins (#)	83
Features	Advanced sensing, DMA, LCD, Real-time clock, Scan interface
UART	2
USB	No
I2C	2
SPI	4
Comparator channels (#)	16

Figure 69, **A MSP430FR6989 Datasheet**

Microcontroller Conclusion

In conclusion, the microcontroller we will be using for this project is the Arduino Uno. The Arduino Uno has the perfect blend of price, usability, functionality, and accessibility that makes it desirable over the other 2 microcontrollers. Some benefits it has over the other 2 boards is the vast variety of modules, lower overall price, power options, programmability, and efficiency.

When compared to the Raspberry Pi, the Arduino Uno does exactly what it needs to do. However, the Raspberry Pi seems to be overkill. It can do the minimum needed but it has extra features such as GPU and CPU that aren't needed. When compared to the MSP430 series, the Arduino Uno has much more support with its parts and is easier to program. The MSP430 requires knowledge of registers and other things to do its functions. The Arduino Uno has many different community and third party libraries and support to make programming simple tasks easier to achieve.

Sensors

Weight Sensor

One sensor that is required for this project is a weight sensor. The sensor is needed to weigh the food based on the size of the pet. A bigger pet will of course require more food. The weight sensor will also need to be compatible with the Arduino Uno microcontroller. The sensor cannot be too big because it needs to be able to fit inside the casing of the Pet Feeder. The sensor also cannot be too small because it needs to hold the food. Many sensors often need a library to work with the Arduino Uno. Luckily due to the resourcefulness of the manufacturer and the community guides, there are plenty of free libraries that allow the Arduino Uno to connect and recognize the inputs of any scales or amplifiers.

For Arduino Uno, there are 2 kinds of Load Cells. One is the rectangular prism shape known as a strain gauge. The other load cell is a flatter, square shape known as a bathroom

scale or half-bridge gauge. Both load cells work identically but are different shapes and dimensions.

HX711 and Bar Strain Gauge
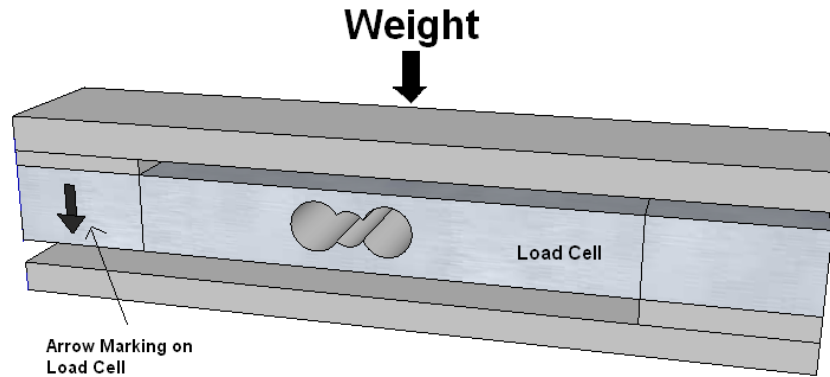
Figure 70, **A Load Cell diagram**

The Load Cell is a component that allows for taking weight measurements. It takes in weight and releases an output voltage signal of the weight. However, this voltage signal is very low (millivolts) and will need to be amplified so it can be read. This is where the HX711 comes in as an amplifier sensor. The HX711 takes in the voltage from the load cell and sends it to the Arduino Uno so it can be calculated. The HX711 also includes analog to digital conversion up to 24 bits.

Due to the way this Load Cell works (using force or torque to calculate weight rather than compression) it must be calibrated and set up. Setting up the load cell requires 2 components. One is the support structure, placed under the Load Cell to hold it and used for base calculations. The other component is the load force, placed above the Load cell that objects are placed on. This requires more work than the half-bridge (bathroom scale) strain gauge, but it allows for custom platforms. However, we only need a container for this project, so the Bar Strain Gauge

Programming the HX711 to the Arduino is simple. Using the HX711 library and the correct layout, we can easily calibrate them together. Calibration is done every time when using a Load Cell to get accurate results and insure the parts are working as intended. A sample diagram provided by electopeak is shown.
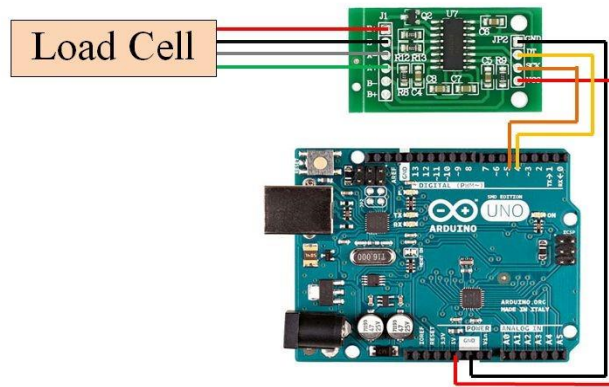
Figure 71, **A Load Cell to HX711 to Arduino Uno circuit diagram**

The Hx711 amplifier is priced at $8 and a 5KG load cell is priced at $3. A total of $11.

- Capacity: 1Kg

- Rated output: 1.0 ± 0.15 mV/V

- Nonlinearity: 0.05%FS

- Repeatability: 0.03%FS

- Creep: 0.1%FS / 5min

- Temperature effect on sensitivity: 0.003%RO/°C

- Temperature effect on zero: 0.02%RO/°C

- Zero balance: ± 0.1%RO

- Input resistance: 1066 ± 20

- Output resistance: 1000 ± 20

- Insulation resistance: 2000MΩ

- Recommended excitation voltage: 5V

- Compensated temperature range: -10 – +50°C

- Operating temperature range: -20 – +65°C

- Safe overload: 120%RO

- Ultimate overload: 150%RO

Figure 72, **A 1KG strain gauge datasheet**

HX711 and Half-Bridge Strain Gauge

Figure 73, **HX711 and Half-Bridge Strain Gauge**

The other kind of Strain Gauge is the half bridge. This module is more square shape and small in length than the bar shaped gauge.
This module will be used over the bar shaped one due to its size. It is the perfect size to place a tray or container on top of it to measure the weight of any pet food that is dispensed on top of it.

UltraSonic Sensor

Another sensor that is required for this project is an ultrasonic sensor. The purpose of this sensor is to check whether or not the container for the food is empty or full. This signal will be displayed with LED lights. The ultrasonic sensor can check the contents of the container by the distance the sound travels. If the container is approaching vacancy then the sound should travel a distance before it hits the other side of the container and goes back to the receiver. If the distance is very short, there should be food in the way of the container and the sound will instantly bounce back to the receiver. This way we can check the capacity of the food.

Of course, another requirement of the ultrasonic sensor is its ability to work with the Arduino Uno. With the simplicity of the Arduino Language and it's libraries, retrieving the result from the sensor is simple and monitoring it is a simple loop. The signal will travel to the Arduino to light up the respective LEDs.

Ultrasonic sensors are generally known for their lower power and easy detection method. Since it relies on sound waves; color, light and transparency don't affect it. The ultrasonic sensor will be in a container to measure its contents, so the range doesn't need to be far and the objects are still-moving: the food and the container.

HC-SR04 ultrasonic sensor

Figure 74, **A HC-SR04 ultrasonic sensor**

One ultrasonic sensor that is compatible with the Arduino Uno is the HC-SR04 ultrasonic sensor. It requires 5V and has a range of 1 inch to 13 feet. However, short ranges will be used more due to the container's size. It comes complete with a transceiver and receiver for all the sound waves. The sensor costs only $1. A design circuit is provided by tutorialspoint.
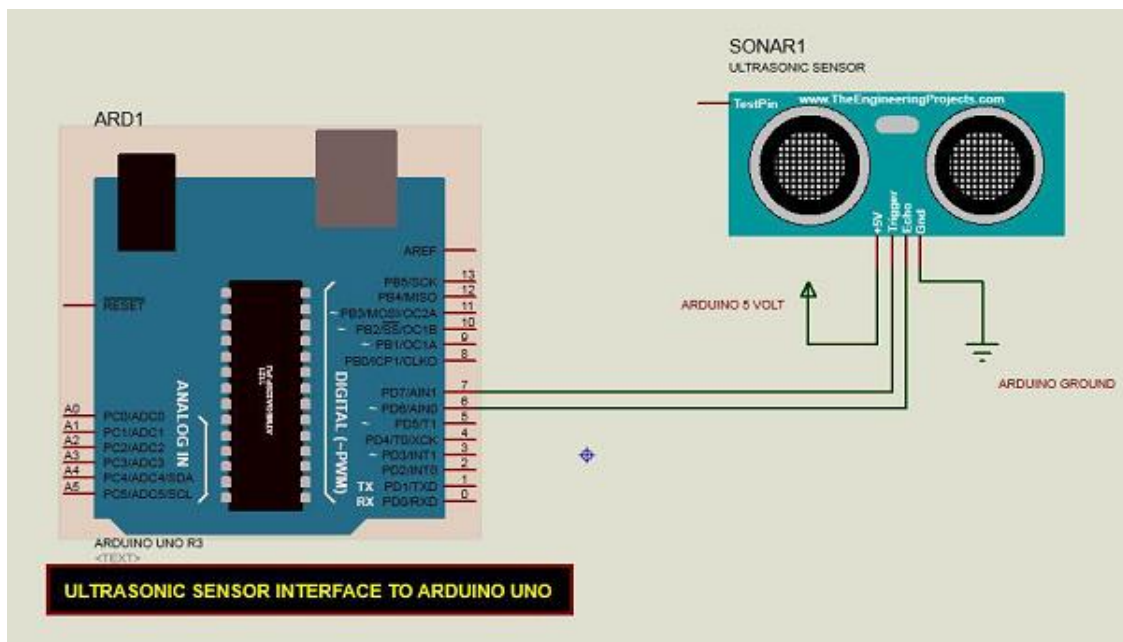
Figure 75, **Arduino Uno connected to HC-SR04 diagram**

Working Voltage	DC 5 V
Working Current	15mA
Working Frequency	40Hz
Max Range	4m
Min Range	2cm
MeasuringAngle	15 degree
Trigger Input Signal	10uS TTL pulse
Echo Output Signal	Input TTL lever signal and the range in proportion
Dimension	45*20*15mm

Figure 76, **A HC-SR04 datasheet**

URM09 Analog Ultrasonic Sensor
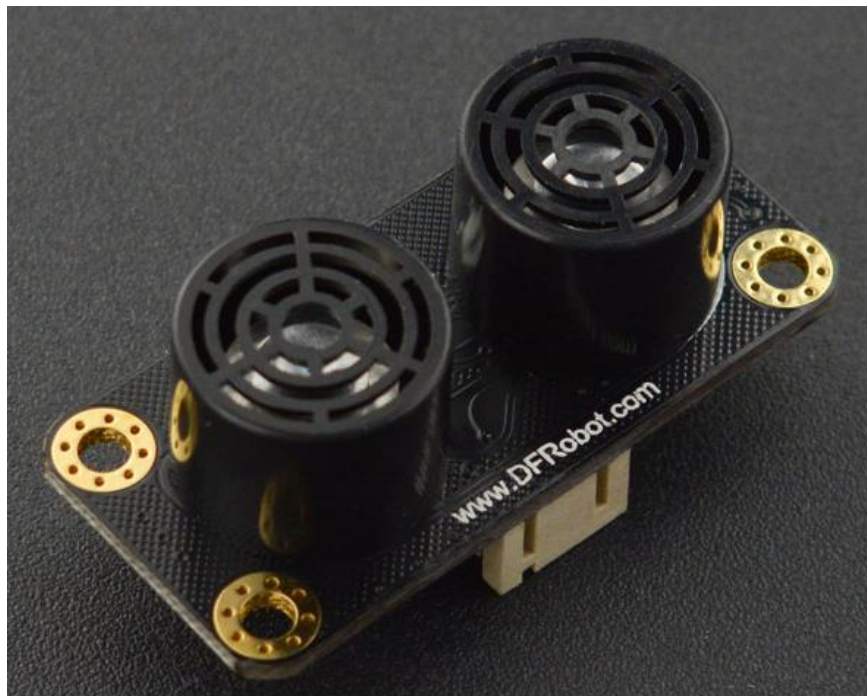
Figure 77, **An URM09 Analog Ultrasonic Sensor**

Another option for an ultrasonic sensor is the URM09. The URM09 is an I2C sensor that supports up to 3 different kinds of ranges: 150cm (59 in), 300cm (118 in) and 500cm (197 in). Its measuring frequency is 50Hz. It is priced at $12.90, 13 times greater than the HC-SR04 sensor. The price combined with the massive community support and projects for the HC-SR04 make it more favorable for being an ultrasonic sensor. Therefore, for this project we will be using the HC-SR04.

Category	Sensors, Transducers Ultrasonic Receivers, Transmitters	
Mfr	DFRobot	☐
Series	-	☐
Package	Bulk ⑦	☐
Part Status	Active	☐
Type	-	☐
Frequency	50Hz	☐
Voltage - Rated	3.3 V ~ 5.5 V	☐
Beam Angle	-	☐
Operating Temperature	-10°C ~ 70°C	☐

Figure 78, **An URM09 Analog Ultrasonic Sensor Datasheet**

Other sensors

There were other sensors that we could have used for checking the contents of the container. However an ultrasonic sensor will suffice for this project due to its shorter range and compatibility with the Arduino Uno.

Radar Sensor

Radar sensors work similar to ultrasonic sensors but they use radio-waves to detect objects with a distance of up to 500 feet. For our project, this is overkill. We only need to detect objects around 2 feet in distance.

LiDAR Sensor

LiDAR (light detection and ranging) sensors work with a light wave rather than radio or sound waves. They can calculate the range of targets by sending and receiving light, by using the time light takes traveling in the air to go and return. It would work just as well, if not better, than the ultrasonic sensor, however it is very costly. A basic sensor starts at around $40. The use of the sensor is very simple in this project, LiDAR is not worth it's price for something that doesn't need to do anything complicated.

IR Sensor

IR (infrared) sensors were the first consideration for this project. IR sensors use 2 lenses: one lense is used to emit the light and the other one is used to receive the reflected light. This would suffice for this project, the IR sensor would work similar to the ultrasonic sensor. If there was food in the path of the IR's laser, then the distance would be a few inches. If there was no food present in the path of the laser, then the container should be nearing depletion. However, there are 2 problems with this approach. Problem 1, the container is clear and see-through so the light might just pass through the container. Problem 2, if the contents are ever in a strange shape, such as a U shape, in the container then the reading will not be accurate.

3.4 Wifi and Mobile

One of the motives of our product is that many people have pets but also have a busy schedule. Many times we may leave our homes and forget to feed our pets, or are gone for long periods of time and feel the need to rush home just to feed them. So one of the ways we want to make this product fix that situation is to give the users of our product the ability to dispense food for their pet remotely. This solution gives us many problems such as needing the ability to connect to the device remotely, and having a way to control the device.

To connect to the pet feeder device we have a few options such as bluetooth, wifi, and zigbee. Each coming with their own complications and benefits. More information of each type will be specified below in the connection comparison section. While there are more ways to connect remotely these are the most popular and easily accessible options.

Another issue is how would a user be able to connect and control the pet feeder device remotely. There are many different ways to control the device such as having a remote control to send signals or having a mobile app that can send signals. We can also have instead of a remote device that sends signals, just control the device from the LCD itself on the device.

Connection options Comparisons

One of the options we have to connect to the pet feeder device is bluetooth. Bluetooth is a very popular method of wireless connection. It is able to transmit data between devices wirelessly, is inexpensive, low energy consumption, and much more. The biggest downfall for bluetooth is its range of usability, bluetooth can be only used at a max of 100 meters.

Another option we have for connecting to the pet feeder is zigbee. Some advantageous features of zigbee are that it is low cost, it is easy to implement and install, and it has a flexible network structure, which means that it can easily be changed and altered. Some downfalls of zigbee are some of the same as bluetooth. It is not fully secure when transmitting data, it is slow transmitting data, and since zigbee doesn't have a lot of end devices yet it also has limited range like the bluetooth technology.

Finally the last option we have considered for our pet feeder is connection over wifi. Some advantages of wifi is that it can transfer data fairly quickly compared to bluetooth or zigbee. It can also transmit a lot more data that a zigbee can in less time. Wifi also has very far reach and allows a very large range compared to the two other options. Some disadvantages of using wifi is that there can still be security issues. Installation might be another problem since wifi can be interfered with by other radio waves and signals around.

So after looking through our options a big thing for us is that we wanted to allow users to connect to our device remotely from anywhere so wifi gave us the range we needed. Even though the other two options were great, they didn't have the range we needed.

Wifi Chip Comparisons
MKR1000:
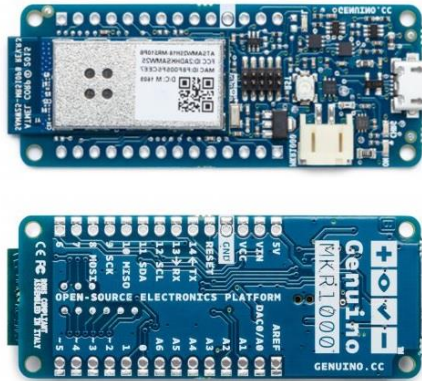
Figure 79, **MKR1000 Wifi Module**

The MKR1000 is a wifi module that I found that was compatible with the Arduino Uno. This board is a little more on the expensive side of the market costing on average $34.99 each. Even though this device is on the pricier side there are some benefits when it comes to selecting this board. Some benefits are that when it comes to developing code for this there is little to no experience required to do so, also this board is specifically created for internet related projects. One problem with this wifi module is that the pins for this specific device are very sensitive and can only handle an input of 3.3V otherwise there can be damage to the board.

Microcontroller	SAMD21 Cortex-M0+ 32bit low power ARM MCU
Board Power Supply (USB/VIN)	5V
Supported Battery(*)	Li-Po single cell, 3.7V, 700mAh minimum
Circuit Operating Voltage	3.3V
Digital I/O Pins	8
PWM Pins	12 (0, 1, 2, 3, 4, 5, 6, 7, 8, 10, A3 - or 18 -, A4 -or 19)
UART	1
SPI	1
I2C	1
Analog Input Pins	7 (ADC 8/10/12 bit)
Analog Output Pins	1 (DAC 10 bit)
External Interrupts	8 (0, 1, 4, 5, 6, 7, 8, A1 -or 16-, A2 - or 17)
DC Current per I/O Pin	7 mA
Flash Memory	256 KB
SRAM	32 KB
EEPROM	no
Clock Speed	32.768 kHz (RTC), 48 MHz
LED_BUILTIN	6
Full-Speed USB Device and embedded Host	
LED_BUILTIN	6
Lenght	61.5 mm
Width	25 mm
Weight	32 gr.

Figure 80, **MKR1000 Wifi Module Specifications**

MKR1010:
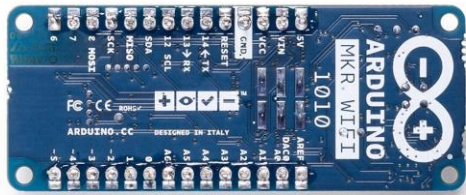
Figure 81, **MKR1010 Wifi Module**

The MKR1010 is similar to the MKR1000, but has a few tweaks to it. This wifi chip also has the capability of connecting to Bluetooth and BLE. It is also compatible with multiple cloud services such as azure, firebase, Blynk, and a few others. This device comes in on the pricier side as well like its sibling the MKR1000, costing about $32.10 each. Even though this board seems to be an upgrade from the MKR1000 I don't understand why it is cheaper. It seems this board does not have as many capabilities when it comes to wifi connection as its sibling the MKR1000, but has many other aspects that are also very useful as stated before.

Microcontroller	SAMD21 Cortex®-M0+ 32bit low power ARM MCU (datasheet)
Radio module	u-blox NINA-W102 (datasheet)
Board Power Supply (USB/VIN)	5V
Secure Element	ATECC508 (datasheet)
Supported Battery	Li-Po Single Cell, 3.7V, 1024mAh Minimum
Circuit Operating Voltage	3.3V
Digital I/O Pins	8
PWM Pins	13 (0 .. 8, 10, 12, 18 / A3, 19 / A4)
UART	1
SPI	1
I2C	1
Analog Input Pins	7 (ADC 8/10/12 bit)
Analog Output Pins	1 (DAC 10 bit)
External Interrupts	10 (0, 1, 4, 5, 6, 7, 8,9, 16 / A1, 17 / A2)
DC Current per I/O Pin	7 mA
CPU Flash Memory	256 KB (internal)
SRAM	32 KB
EEPROM	no
Clock Speed	32.768 kHz (RTC), 48 MHz
LED_BUILTIN	6
USB	Full-Speed USB Device and embedded Host
Length	61.5 mm
Width	25 mm
Weight	32 gr.

Figure 82, **MKR1010 Wifi Module Specifications**

ESP8266:
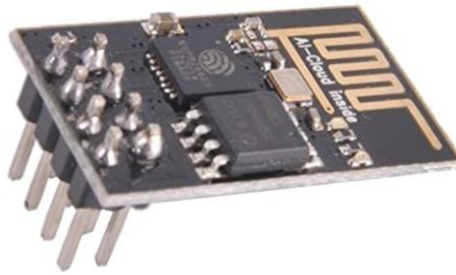
Figure 83, **ESP8266 Wifi Module**

The ESP8266 wifi module is also another one that I found is compatible with the MCU we have selected. This board is one of the older boards on the market but still has the capabilities to connect our device to the internet. This module cost on average $4 each. Some benefits of this board is that it is cheap and easy to use, and has lots of previous public code and libraries to help with development. Some cons are that it does not seem as powerful as some of its competitors but it gets the job done.

Categories	Items	Parameters
Wi-Fi	Certification	Wi-Fi Alliance
	Protocols	802.11 b/g/n (HT20)
	Frequency Range	2.4 GHz ~ 2.5 GHz (2400 MHz ~ 2483.5 MHz)
	TX Power	802.11 b: +20 dBm
		802.11 g: +17 dBm
		802.11 n: +14 dBm
	Rx Sensitivity	802.11 b: –91 dbm (11 Mbps)
		802.11 g: –75 dbm (54 Mbps)
		802.11 n: –72 dbm (MCS7)
	Antenna	PCB Trace, External, IPEX Connector, Ceramic Chip
Hardware	CPU	Tensilica L106 32-bit processor
	Peripheral Interface	UART/SDIO/SPI/I2C/I2S/IR Remote Control
		GPIO/ADC/PWM/LED Light & Button
	Operating Voltage	2.5 V ~ 3.6 V
	Operating Current	Average value: 80 mA
	Operating Temperature Range	–40 °C ~ 125 °C
	Package Size	QFN32-pin (5 mm x 5 mm)
	External Interface	-
Software	Wi-Fi Mode	Station/SoftAP/SoftAP+Station
	Security	WPA/WPA2
	Encryption	WEP/TKIP/AES
	Firmware Upgrade	UART Download / OTA (via network)
	Software Development	Supports Cloud Server Development / Firmware and SDK for fast on-chip programming
	Network Protocols	IPv4, TCP/UDP/HTTP
	User Configuration	AT Instruction Set, Cloud Server, Android/iOS App

Figure 84, **ESP8266 Wifi Module Specifications**

ESP32:

Figure 85, **ESP32 Wifi Module**

The ESP32 is the successor of the ESP8266. This board is a little more expensive than the ESP8266 costing on average about $10 each. This board has a Dual core processor and also has the capability for bluetooth 4.2 and BLE. This board seems to have all the capabilities as the ESP8266 but a little more just added on top such as faster processor, more pins and channels and bluetooth capabilities.

Parameter	Condition	Min	Typical	Max	Unit
Operating frequency range [note1]	-	2412	-	2484	MHz
Output impedance [note2]	-	-	*note 2*	-	Ω
TX power [note3]	11n, MCS7	12	13	14	dBm
	11b mode	17.5	18.5	20	dBm
Sensitivity	11b, 1 Mbps	-	−98	-	dBm
	11b, 11 Mbps	-	−89	-	dBm
	11g, 6 Mbps	-	−92	-	dBm
	11g, 54 Mbps	-	−74	-	dBm
	11n, HT20, MCS0	-	−91	-	dBm
	11n, HT20, MCS7	-	−71	-	dBm
	11n, HT40, MCS0	-	−89	-	dBm
	11n, HT40, MCS7	-	−69	-	dBm
Adjacent channel rejection	11g, 6 Mbps	-	31	-	dB
	11g, 54 Mbps	-	14	-	dB
	11n, HT20, MCS0	-	31	-	dB
	11n, HT20, MCS7	-	13	-	dB

Figure 86, **ESP32 Wifi Module Specifications**

Arduino Uno Wifi Rev 2:
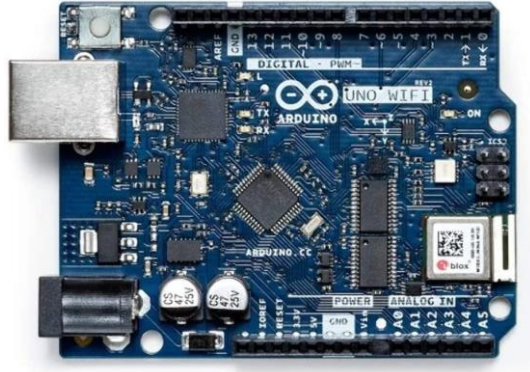
Figure 87, **Arduino Wifi Rev 2 MCU with integrated wifi**

This Board is very different from the precious boards that have been presented because this is the Arduino Uno wifi REV2, it is not just a wifi module like the rest it is a microcontroller with integrated wifi and bluetooth. Even though this seems like a nice one stop shop it comes at a pretty high cost of an average of $45. Also this board does not have all the libraries for TCP connection which may cause a problem down the line. There are community made libraries that might fix the issue but, the issue must be taken into account.

Microcontroller	ATmega4809 (datasheet)
Operating Voltage	5V
Input Voltage (recommended)	7 - 12V
Digital I/O Pins	14 — 5 Provide PWM Output
PWM Digital I/O Pins	5
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	48 KB (ATmega4809)
SRAM	6,144 Bytes (ATmega4809)
EEPROM	256 Bytes (ATmega4809)
Clock Speed	16 MHz
Radio module	u-blox NINA-W102 (datasheet)
Secure Element	ATECC608A (datasheet)
Inertial Measurement Unit	LSM6DS3TR (datasheet)
LED_BUILTIN	25
Length	68.6 mm
Width	53.4 mm
Weight	25 g

Figure 88, **Arduino Wifi Rev 2 MCU with integrated wifi Specifications**

Wifi chip Selection

After looking through multiple wifi chips and comparing their pros and cons, it seems like one main issue here is lacking libraries to help with development. After careful

consideration and thought We have selected the ESP8266 as our wifi chip because there are too many uncertainties when we don't have the right libraries to develop the wifi. Also being new to wifi development we want something that will surely have plenty of documentation to be able to make it work, plus it was the cheapest chip available making production easier when it comes to cost.

Wifi chip Pinout

Figure 89, **ESP8266 Pinouts**

Here we see that the ESP8266 has 8 different pinouts one is for the power supply, 3.3V VCC, there is one for ground. Then there are the two pins for communication between the wifi chip and the arduino uno the Tx and Rx. There are also two pins for the settings for resetting and channel enabling. Finally we have the GPIO pins 1 and 2 for general purpose and controls. These pins will most likely be used for more data control.

Wifi chip Features
- Low cost, compact and powerful Wi-Fi Module
- Power Supply: +3.3V only
- Current Consumption: 100mA
- I/O Voltage: 3.6V (max)
- I/O source current: 12mA (max)
- Built-in low power 32-bit MCU @ 80MHz
- 512kB Flash Memory
- Can be used as station or Access Point or both
- Supports Deep sleep (<10uA)
- Supports Serial communication hence compatible with many development platforms like Arduino
- Can be programmed using Arduino IDE or AT-commands or Luna script

We can see from the specs of the device that it is highly compatible with arduino and that it does not consume much power, which is good for the power saving aspect.
Wifi Development

To begin the wifi development we must connect the wifi module to the main arduino microcontroller. The connection should look similar to the image below.
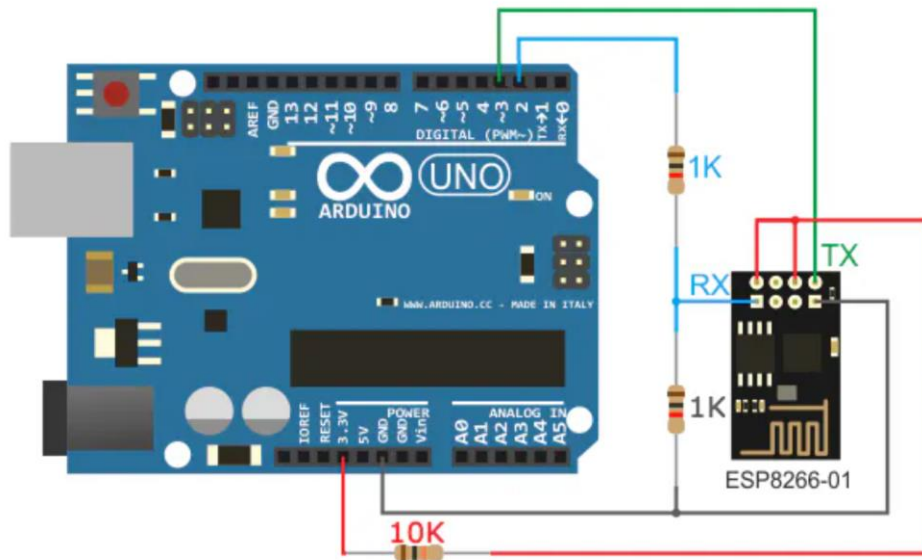
Figure 90, **ESP8266 connection to Arduino Uno**

To connect the wifi module to the arduino microcontroller we will need two 1K resistors and one 10K resistor this is to create a voltage drop from the microcontroller and the wifi module, since the wifi module is sensitive and can only operation on a voltage of 3.3V. If we were to use the full 5V from the main MCU it would burn out the wifi module and damage it causing it not to work over time.

After finally connecting the wifi module to the main MCU we will have to develop code that allows the ESP8266 to identify itself to the MCU and connect to the internet wirelessly. To do that there must be a lot of information given so it knows what wireless internet to connect to and permissions such as a password to access it. Right now we have only worked on a hard coded version of this but would like to be able to make it more robust for users to scan for available internet around and connect to it without having to touch any of the code.

To test if the wifi module is set up correctly we will be using an app called the TCP client to see if we can send basic commands and see if the board will respond. If the board responds we can then go ahead and start developing the main mobile application for our pet feeder device. In the next section there will be a simplified version on how we will plan on developing the mobile app and testing it. Later in the "Design" Section of the product there will be a very in depth explanation on how we plan on testing and developing our mobile app and wifi module to make sure everything is working and are up to the specified engineering standards along with our own standards for the product.

```
#include <ESP8266WiFi.h>

void setup()
{
    Serial.begin(115200);
    Serial.println();

    WiFi.begin("network-name", "pass-to-network");
}
```

Figure 91, **Wifi setup code snippet**

From the image above we can see that we will need to import the "ES8266WiFi.h" library to be able to connect to any wireless internet. Another important part of the code here is "WiFi.begin("network-name", "pass-to-network")" because this is the actual code where the wifi chip attempts to connect to a wireless network. So two crucial pieces of information we will need are the network name and password. This is also just the initial set up to connect to the internet but has nothing yet to do with passing information and commands to control the MCU.

Figure 92, **AT Command List**

Commands	Description	Type
AT+RST	restart module	basic
AT+CWMODE	wifi mode	wifi
AT+CWJAP	join AP	wifi
AT+CWLAP	list AP	wif
AT+CWQAP	quit AP	wifi
AT+CIPSTATUS	get status	TCP/IP
AT+CIPSTART	set up TCP or UDP	TCP/IP
AT+CIPSEND	send data	TCP/IP
AT+CIPCLOSE	close TCP or UDP	TCP/IP
AT+CIFSR	get IP	TCP/IP
AT+CIPMUX	set multiple connections	TCP/IP
AT+CIPSERVER	set as server	TCP/IP

After connecting the wifi module to the internet and the MCU we have to test it to see if it is set up correctly. We will have to also make sure the MCU recognizes the wifi module and one way we can do that is possibly flashing the ESP. You can find more about this looking up Flashing ESP module.

Once all conditions above are met we can then test the ESP8266 wifi module to see if it is working. One way to do that are AT commands this allows

Figure 93, **TCP Client Demo**

```
Add a item
Name
  Test
IP
  192.168.4.1
Port
  80
      Add        Cancel
```

us to make the wifi module do specific things and return something. The way you would do AT commands is downloading an app called "TCP client" for whatever platform you are using such as Android or IOS. You have to connect to the ESP88266's IP address and connect to port 80, which is the main port used for communication. When you are in the app this is how it should look like when you are trying to connect to the device for the first time. Once all information is correct, add the device and then we can start sending the at commands to see if the board is properly set up. This is just an example so information may be different from board to board such as the port number and IP address this will have to be found out by the user manual.

Figure 94, **Void Loop code snippet for main functions**

```
void loop() {}
```

This void loop is the meat of the code where we can call our created functions to control the MCU. Inside the brackets is where we can call our own made functions, or premade functions to control the MCU. This Function is a loop so it will run indefinitely waiting for commands and executing them. This function will run until the code is manually told to shut down or has to be restarted again. This function will also be the access point for our mobile app. We want to create a mobile app that can send the commands that we created to dispense food or set timers in the MCU.

Mobile APP Environment

Now that we have figured out how to connect our MCU to the internet, the next issue is how to give users the ability to control the pet feeder remotely without having to be at home. Since smartphones seem to be very popular and everyone has access to some smartphone, we believe making a mobile application would be the best solution to this problem. When it comes to mobile apps there is a lot to consider such as where to develop this app, what platforms should be go for, how the app should look.

After some consideration we decided to develop an application for Android because they are more user friendly for developers and made it easier to test for individuals who own an android or even using an Android emulator. On top of all this most integrated development environments for android are free and do not require any extra certifications. After deciding on Android as our platform we looked at a few different IDEs such as Android Studios, Visual Code Studios, Eclipse, etc. but after looking at them we decided to go with Android Studios because most of us has had some experience with that IDE and it allows us to graphically see our app as we are developing it which is useful seeing how everything is on the screen.

As we can see in the image below there is a split screen that we are able to view how we have developed the app so far on the screen. The visual image of the app updates in real time as you change and update your code, this is especially helpful for finding bugs and mistakes in your code because the moment something is wrong the error can be seen on the screen or it will just disappear if your code did something to do that. While the visual aspect is a great debugging tool it can be used for much more than just that. On the visual side of the split screen we can also add buttons, text boxes, labels, and so much more. As we are adding those things to the visual aspect the code is also updated to have everything that was added on the visual part. From there on the code side we can use some of the built in functions to get data from the app or have it execute commands based on if buttons were clicked or not. To test this we will need an android phone or use a downloaded android emulator, if we use the emulator we can use it right in the android studio IDE but if we use a cellular device we can either connect a USB and download it on the phone or download using an APK file.
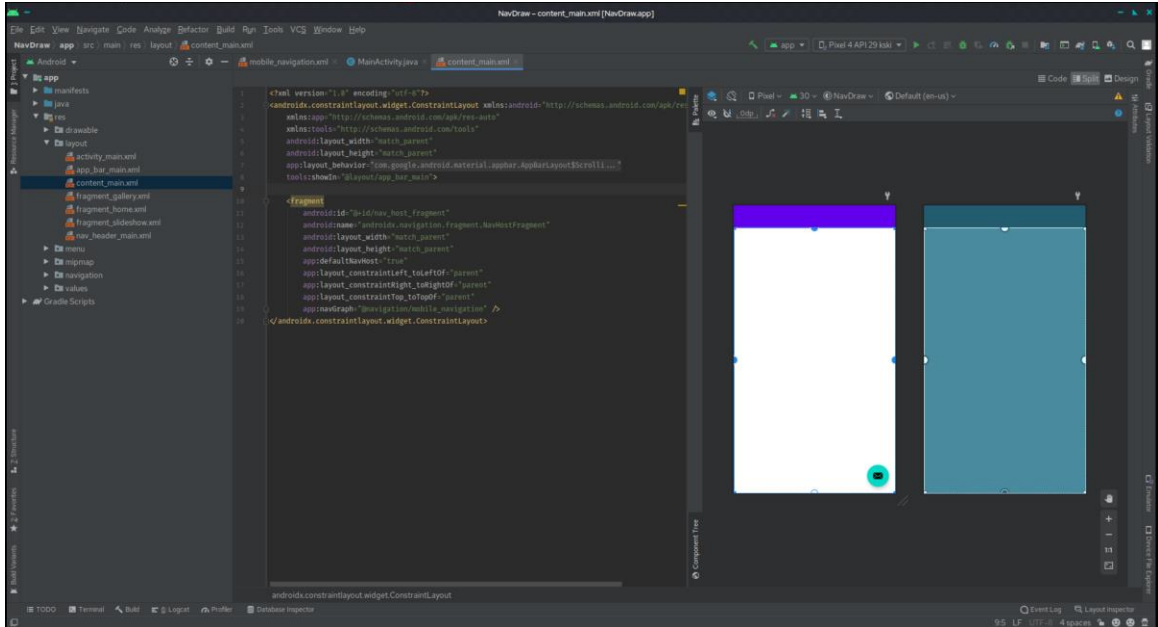
Figure 95, **Android Studio code and visual representation**

Mobile APP Development

The language that we decided to use to develop the mobile app is going to be kotlin. The reason being that we have the most experience with kotlin when it comes to developing a mobile app.
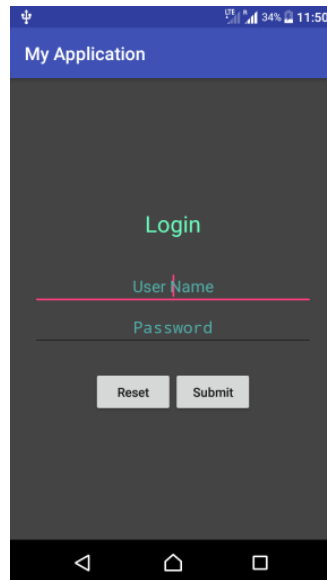
Figure 96, **Sample Login Page**

The first thing that we would like to have on the mobile app is a login screen that allows only users that we have registered being able to use the app, this way it would be easier to help clients with accounts and trouble later on. We would also want a register page for people who do not have an account already.

Once the client has logged in it should go directly to the homepage where there would be three to four buttons for manually dispensing meals and the sizes of the meals. Since there are various pet sizes and weights there are recommended serving for each of them and based on that information we can have a snack, small, medium, and large portioning based on size. The way these portionings are determined is by a weighing scale the food will continue to dispense until the weight sensor has gotten enough for that portion then it will notify the gate to close and stop dispensing food.
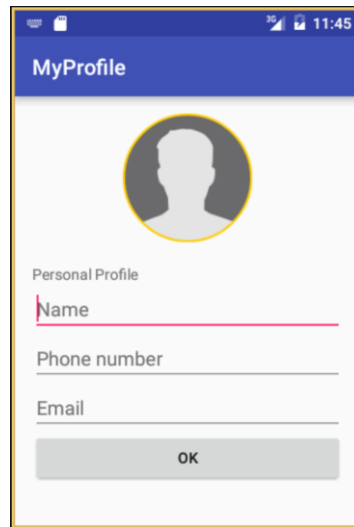

Figure 97, **Sample Account Page**

We would also prefer to have an accounts page with all the information of the client and what pet of the client this should be able to be updated at any time the client would like to change it. On this account page the client should also have access to all of the devices that they own if they have added it to the app. So a client can have multiple devices in their home for various pets that they may have. When the client accesses each of their devices is where they can enter in information for their pet and configure the settings for each device.

Another cool aspect we would like to add to the pet feeder mobile app is a last time fed display. This would be very useful for any pet owner to know when was the last time they fed their pet manually or by the times, since everyone is usually one a busy schedule it would be nice to have a reminder if your pet was fed or not just in case you forgot. Overall we want the mobile application to be as intuitive as possible, making it very easy to use. We also want to make it robust for upgrades on information on controlling multiple devices.

This section is a quick and brief demonstration on how we plan on developing our application. We have not yet talked about it in a lot of detail yet. The details can be found more in depth in the Design section of our paper, these are just samples.

4. Design Constraints and Standards

This section will cover all standards that our project will like to abide by and all constraints that will affect our overall product. We will talk more in depth of specific constraints that have affected the team, and standards that should be followed to create the best product possible.

4.1 Standards

Our daily lives are constantly changing because of the engineering development of products and technologies manufactured worldwide. The development of our lives is due to the use of personal devices, sophisticated transportation, and the ever-changing technology in general. Despite the speed of change of these technologies, they all have to meet certain standards before they become available in the market. These standards are set by different organizations, agencies, and governments worldwide in order to assure the quality and safety of the products before they get into the hands of the consumer.

For each single engineering product, there are dozens of different standards that the manufacturer has to meet. These standards are across the board, they are not typically specified to every manufacturer since a single product can be produced by different manufacturers. Therefore, these manufacturers have to meet the minimum standards before their product makes it to the consumer. Evidently, each manufacturer can set their own standards. However, these standards that are set by a specific manufacturer have to exceed the overall standards set by the governing bodies. This is also very important to the consumer in order to get the products are of high quality and are safe to operate. . There are many governing entities when it comes to setting standards, the major organizations ISO (International Standards Organization), IEEE (Institute of Electrical and Electronic Engineers), ANSI (American National Standards Institute) and OSHA (Occupational Safety and Health Administration).

Occupational Safety and Health Administration

The Occupational Safety Health Administration, OSHA, is the government agency that regulates workplace standards for labor safety. All companies and employers have to abide by OSHA standards in order to assure a safe work environment for their employees. Most consumer products have to follow OSHA standards in order to label their products as not hazardous. This is critical to our project since it will be used as a household item and should not cause any hazards that can harm anyone. This is also very important to the project designers since we will be testing the project either at home or in the lab and we have to follow the safety procedures. Since OSHA standards are followed at the labs, we should also follow these standards during all the aspects of testing and implementation phases of our project. During testing, we will be using an electrical multimeter to measure the voltages and currents of different components of the circuit. Therefore, we should be aware of the potential of electric shock or injury as outlined by OSHA standard [30] below.

Current	Reaction
1 Milliampere	☛ Perception level, a faint tingle
5 Milliamperes	☛ Slight shock felt, not painful, but disturbing ☛ Average individual can let go ☛ Strong involuntary reactions to shocks in this range can lead to injuries
6-25 Milliamperes (women)	☛ Painful shock ☛ Muscular control is lost
9-30 Milliamperes (men)	☛ Freezing current or "let go" range
50-150 Milliamperes	☛ Extreme pain ☛ Respiratory arrest ☛ Severe muscular contractions* ☛ Individual cannot let go ☛ Death is possible
1,000-4,300 Milliamperes	☛ Ventricular fibrillation (the rhythmic pumping action of the heart ceases) ☛ Muscular contraction and nerve damage ☛ Death is most likely
10,000+ Milliamperes	☛ Cardiac arrest ☛ Severe burns ☛ Probable death

Figure 98, **Effects of electric current in human body Source: OSHA Standards [30]**

The National Electrical Manufacturers Association (NEMA)

The National Electrical Manufacturers Association (NEMA) standards regulate electrical enclosures for protection against any accidental hazard. NEMA sets rating for enclosures for protection against personal access to hazardous parts, these enclosures are rated to provide protection against any liquid from penetrating the electrical components enclosure.

NEMA Rating	Rating Specifications
Type 1	• Indoor use • Provide degree of protection to personnel against hazardous parts • Provide degree of protection of the equipment inside the enclosure against ingress of solid foreign objects (falling dirt)
Type 2	• Same rating specifications as Type 1 • Provide degree of protection with respect to harmful effects on the equipment due to the ingress of water
Type 3	• Same rating specifications as Type 2 • Provide a degree of protection against damage by the external formation of ice on the enclosure
Type 3R	• Same rating specifications as Type 3 • Indoor and Outdoor Use
Type 3S	• Same rating specifications as Type 3R • External mechanisms remain operable when ice laden
Type 3X	• Same rating specifications as Type 3R • Provides a degree of protection to against corrosion

Figure 99, **NEMA NE-NEMA-vs-UL-012314 rating specifications**

Coding Standards

IEEE 1008-1987 Standard for unit testing this is especially important for the mobile aspect of our project because there are many moving parts. Unit testing is a method of testing individual parts of a large program to make sure each of the parts are working well on their own with test data. We can enter in test data and see the output data and make sure it is what we are expecting, that is how you unit test a piece of code. There are going to be a lot of buttons for connection control and editing so every button must be tested separately to know if they are working properly as a single unit before it can be integrated into the project as a whole so the standards on how to unit test will help debug and develop our mobile application.

ISO/IEC/IEEE 24748 Standard focuses on software development planning this is also very important when it comes to developing the mobile application. To be successful and efficient when developing software there must be enough research and planning done to get it done as fast as possible. We have not one a great job so far due to the amount of work being split between the group members, but during some of the meetings we have discussed what we wanted the application to be able to do and how it should look. So that planning helped a lot with the research needed to be done on how to achieve both of the design and capabilities. We will have to work harder to follow the standard on how to develop the mobile application to be more credible in our work.

IEEE/ISO/IEC P24748-6 Standard is one on system and software integration which has to do a lot with our mobile application connecting and communicating to the pet feeder. It will also focus a lot on our wifi, sensors, and display setups. We will want to follow this standard to make sure there is a secure connection and that everything is integrated correctly so there can be no harm to the user. This will also help with the life expectancy of our product. If everything is integrated correctly and tested thoroughly then it will boost our safety and life expectancy standards as well.

IEEE 802E-2020 Standard is mainly about privacy and security and that is something that is extremely important to us. Since today's world is mainly technology based a lot of our information is stored on the web somewhere or in a database somewhere. A lot of the information is sensitive information that can be used against us in many ways such as identity theft, banking information, residential area, and much more. We want to have a very secure mobile application because when getting the device if the owner would like the remote access and control of our device they will have to give us some information signing up on the application, and we would like to try our best to keep all their information secure.

IEEE/ISO/IEC 14764-2006 focusses on maintenance while this does not yet fully apply to our project quite yet we would like to follow these standards to maintain the mobile application and automatic pet feeder to meet the needs of our clients. We would like to create something that is very robust and is always growing, and changing. To do this lot of the code and technology used will need to be maintained to perform faster and better.

4.2 Constraints

Economic Constraints

This year, the most significant economic constraint for most people is the effect of the pandemic. Because of covid-19 millions of people lost their employment, students are one of the hardest hit populations. Since most economic sectors were closed for certain periods, it was very difficult for students to find employment. This is even more difficult for students since most couldn't find employment due to the national economic crisis. In addition to that, the colleges have restricted occupancy, so most students are studying from home. With most economic sectors suffering from the effect of the pandemic, it was very difficult to find sponsors who would be willing to sponsor any project unless it's their own project or area of research. Due to this, we had to improvise and design a low-cost pet feeder and still produce a good, efficient design. Furthermore, and because of the financial constraint, we don't have the privilege of ordering too many parts for testing. We have to purchase a limited number of parts and make sure that we design a functioning product. We started the project without having a specific set budget, but we were diligent during research in order to purchase low cost but efficient parts.

Time Constraints

It's one year now that the whole world is under the effects of the pandemic, the impact didn't spare anyone or any sector. Designing a project under these circumstances is especially difficult for multiple reasons. Most students have not physically attended any in classes this whole time, virtually learning is manageable as long as it doesn't involve any hands-on experiments or laboratories. Designing and building a project is also difficult given the lack of face to face interaction with the coordinators, and the inability to have access to the laboratory test equipment. All these limitations make things extremely hard, time management under these circumstances is difficult.

The project work is taking most of our time, leaving little time for the other classes and personal life. We have many deadlines to meet and too much to accomplish but not much time. We have to balance writing the page counts to meet to set deadlines while we also have to research and check the parts needed for the project. For the parts acquisition, it is mostly difficult to order the parts in order to receive them in a timely manner because of the shutdowns that are still happening in some parts of the world.

Working and Studying from home also puts some limitations on when we get to work on the project due to having different responsibilities while being home and away for school. Being around family means there is more you will have to take care of due to just being around so there are times where you physically and mentally can not work on the project due to the people around you. Sometimes the responsibilities you have while at home are a lot more time consuming than when you are away at work or away at school on your own.

Safety Constraints

Safety constraints restrict the practice or implementation of certain designs. For example, our design uses electricity. There, this device can be very dangerous. It deals with many electrical components that can be a hazard. If the wrong power is supplied the pet feeder may fail to work or even cause electrical hazards such as sparks or flames. For this reason it is strongly advised that anyone else operating this device reads the requirements for this device. Failure to do so may result in hazards. While the operating currents and voltages are relatively low, caution should still be stressed. All electrical points are to be covered and insulted to help prevent any shock. All electrical components should also be properly grounded.

Another safety constraint is any collisions or impacts on the pet feeder. As previously mentioned above, there are many electrical parts such as LCD, sensors, and the microcontroller with various wires used to power them. Any impact or collisions from pets or humans may result in them becoming unplugged or loose. This device should not be overfilled with food products that allow for it to topple and fall over nor be placed on any uneven surfaces.

There are also some moving parts that may result in a hazard. Dispensing the food from the pet feeder container requires the gate to open and close for proper amounts of food. The force used to close the gate is not very strong but it still requires a motor that can cause pinch points resulting in injury.

Health and Ethical Constraints

Health and Ethical constraints restrict the nature and benefit of a design, such as causing harm, illnesses, or hazards due to sickness or lack of wellbeing. Health constraints arise in this design because this device does require it to be filled with food products. The container should be kept clean and washed often to prevent any disease or bacteria that could result in the user, the pet(s), to become ill. It is the responsibility of the owner to provide the device with solid pet food that is consumable for their pet.

Manufacturing Constraints

Manufacturing Constraints restrict the components of a design that can prevent it from being created. In our case, the manufacturing constraints are our parts. Many of our parts are very cheap and therefore can be unreliable in the long run. They function very well on their own for simple testing however if this design requires constant use and might be placed in harsh environments and exposing the pet feeder to elements such as sunlight, some problems may arise. The low quality of certain parts may result in their failure to properly do their tasks such as sensing or dispensing.

5. Design

Will cover all individual Designs of each component.

5.1 Schematic Overview

Circuit Design

*Figure 100*, **Technology Circuit Mapping Diagram**
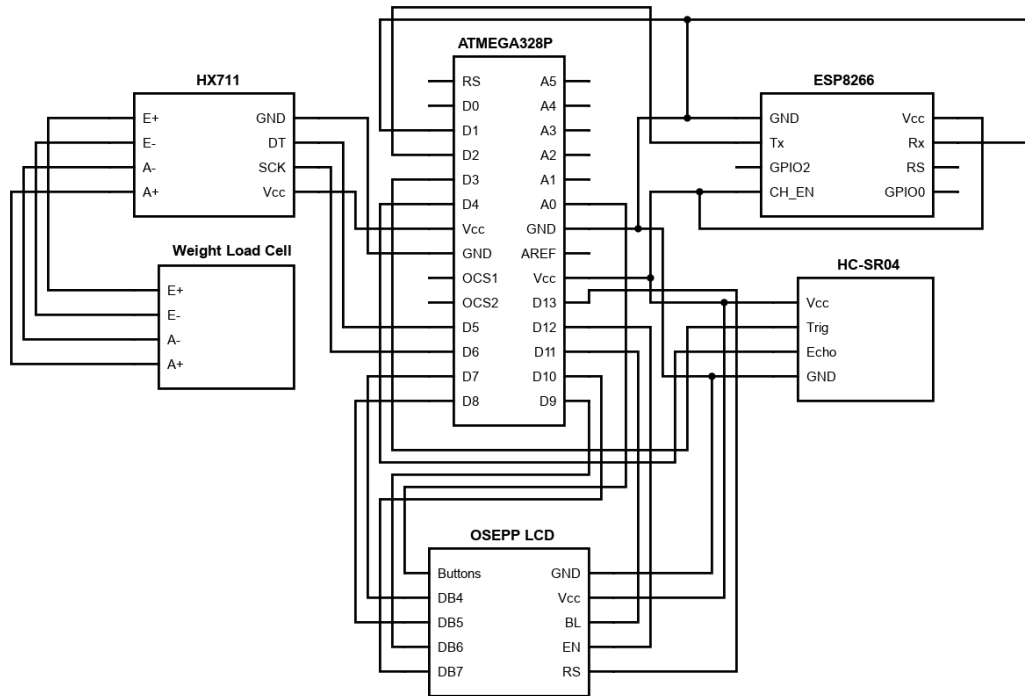
The image above is the entire circuit mapping of all of the components connected to the Arduino. We can see that the ESP8266 which is the wifi module uses up digital pins 1 and 2, while also using the 3.3V Vcc and the GND pins.

The LCD display is using digital pins 7 through 13, this is due to the amount of information and signals needed to display. The LCD is also using the same GND and 3.3V Vcc pins as the ESP8266.

We then have the HC-SR04 which is the ultrasonic sensor to measure the food level of the automatic pet feeder. This device uses digital pins 3 and 4, while also using the same Vcc and GND pins as the other two previous devices.

Finally we have the HX711 which is an amplifier connected to the weight load cell which together makes the weight sensor to weigh out the food portions. The amplifier is using the digital pins 5 and 6, while also using the second GND pin and the 5V Vcc pin. As we can tell from the image the digital pins can not be shared because of possible conflicting information.

Figure 101, **Custom Pet feeder Side View Schematic**
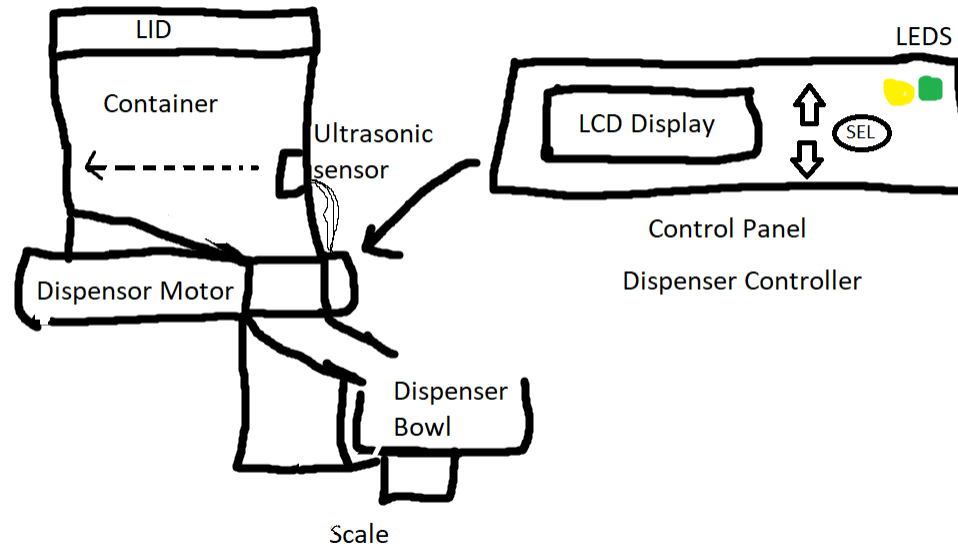
The initial design for the pet feeder is very simple. Starting from the top we have the lid, this will open and close to allow the user to pour solid dry food in. Next is the container. We have no specific size in mind for the container, but it should be able to hold more than 20 cups, which is the minimum for other on-sale pet feeders on the market. The container will also be see-through, so the user can physically see the contents in the container. The container should be slanted so food does not pile up and can slide down the chute. The container will also house the ultrasonic sensor. The sensor checks the contents level and passes the information down to the microcontroller.

Below the container is the dispenser controller. This is what is used to open and close the food dispenser, stopping the food from falling. The controller is operated by the motor, which will open and close it. The motor also supports the weight of the structure so it's equally distributed. The motor will be powered by batteries and does not need to be near the Arduino Uno microcontroller.

Next to the dispenser controller is the Control Panel. The control panel houses everything the user will need to interact with: the LCD display, the up and down buttons, and the contents LEDs. The LCD display shows information such as the selected pet size and the manual dispensation options. The up and down buttons move the LCD's cursor and they can select their choice. The LEDs represent the level of content based on the readings from the ultrasonic sensor: green is ok, yellow is caution.

The last part of the pet feeder is the dispenser bowl. This will hold all of the food dropped and allows the pet to eat. It will be removable so it can be washed and sits right on top of the weight scale. The weight scale will send a signal to the microcontroller to close off the dispensing.

5.2 User Interface and Control

Menu

User Interface and Control General Breakdown:
- LCD Display/Menu
  - User Controlled
    - Selection of portion size: small, medium, large (range of dispensing)
    - Scheduling of dispensing
    - Dispensing of food manually
  - Vitals
    - Status of food level in tank (checked after dispensing)
    - Wifi connection status
    - Power level
    - Portion weight calibration
- Button Control
  - Up/Down buttons to scroll menu
  - Select button to select menu option

The menu will follow the interface labeled above and use the buttons, up and down, to cycle through the menu, and the select button to enter that menu option. The image below demonstrates basic operation of the menu.
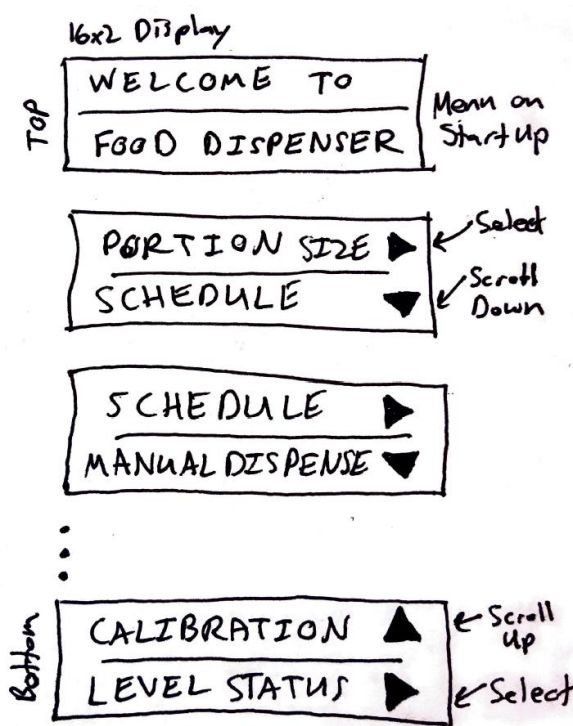

Figure 102, **LCD Menu Visual**

LCD Hardware Implementation
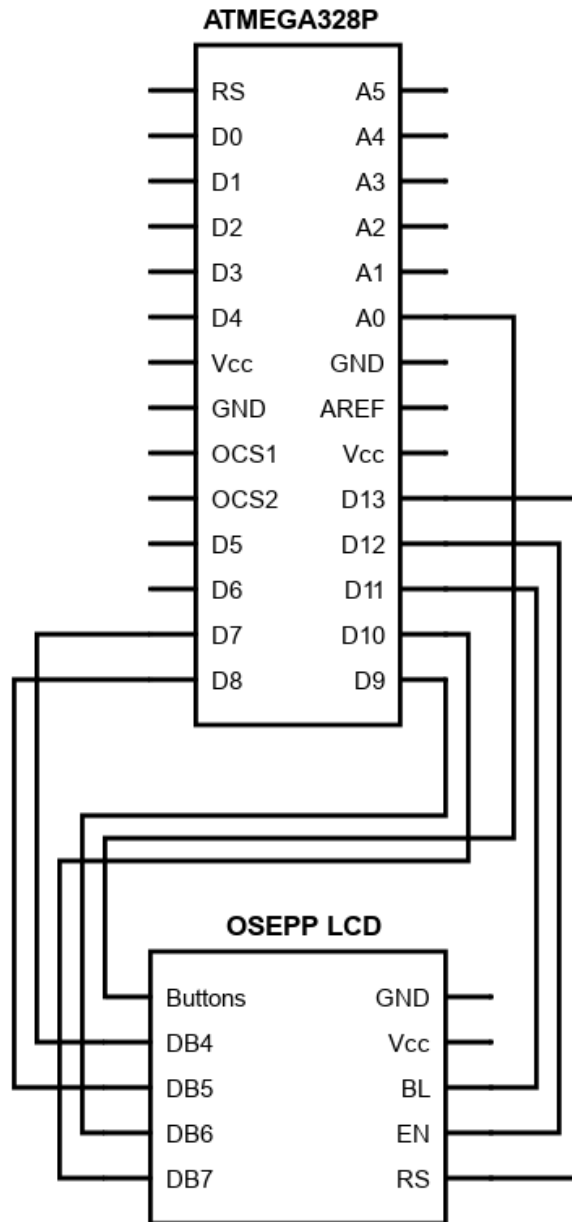
Figure 103, **LCD Circuit Mapping Diagram**

Breadboard connections are not available to show at the moment due to parts still in shipping due to COVID. Shipping has been slow along with production. Breadboard test images and research will be added by the next paper due date as the parts should arrive before that date. The basic schematic in this section demonstrates how this breadboard connection will work.

Figure 104, **LCD Flow Chart**
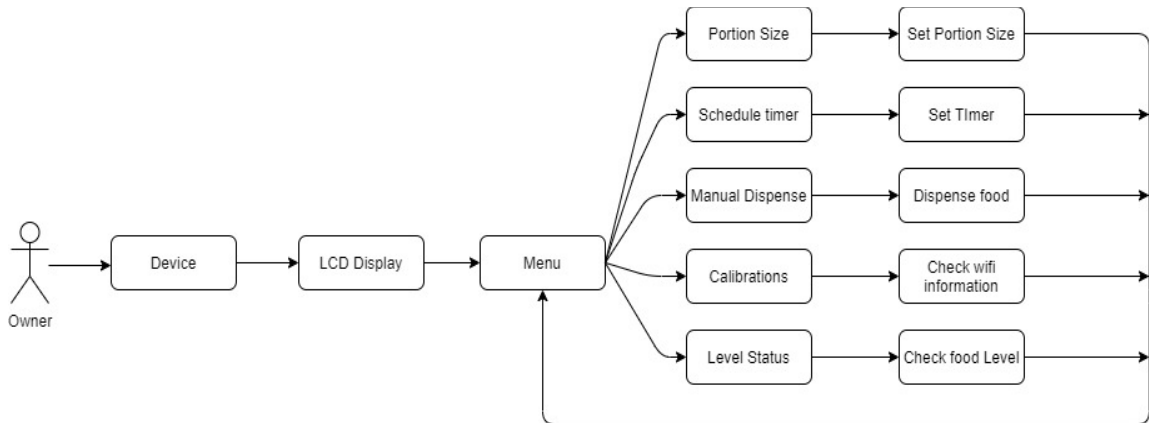
Here we have a flow chart for the LCD display and the control menu. Along with the mobile application pet owners will also have the ability to control the automatic pet feeder through a LCD display that is right on the physical device.

We can see the flow chart starts off with the client or we can also call the pet owner. They will then need to acquire one of our automatic pet feeders. Once that is complete they will have to plug in the pet feeder and turn on the device. The device will then turn on with a welcoming message for a little while then it will automatically proceed to a menu. The menu will have a few control options that the user will then be able to use to control the pet feeder. To scroll through the options there will be an up, down, and select button.

The first menu option will be the portion size, the user will be able to set portion sizes such as snack, small, medium, and large for their pet. The sizes will be calculated based on recommended meal sizes for pets.

The second option will be the Schedule timer option. This option is for setting a timer to dispense food for your pet at specific times throughout the day. This is especially useful for pet owners alway on the go, and/or forgetful. This feature can also be useful for owners who are occasionally late, because it happens every once in a while.

The third option is manually dispensing food, this is for manual use if users would rather keep their pet for an irregular eating time. It can also be very useful if your pet is behaving well and you would like to just give them a little snack for a treat.

We then have the Calibration options, which is like a settings option where you can view and edit information of the device. This can be used to look at your information on the device, the wifi, and much more.

Finally we have the level status which will allow you to see the status of your food level within the pet feeder. Once you have selected your option and finished your command you will be directed back to the menu screen where you can choose a different option and complete another command if needed if not, after some time the LCD will go to sleep until the next time it is in use.
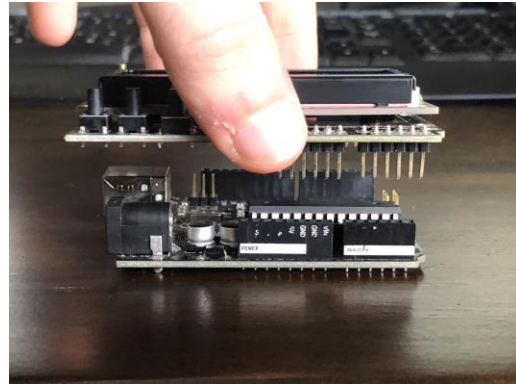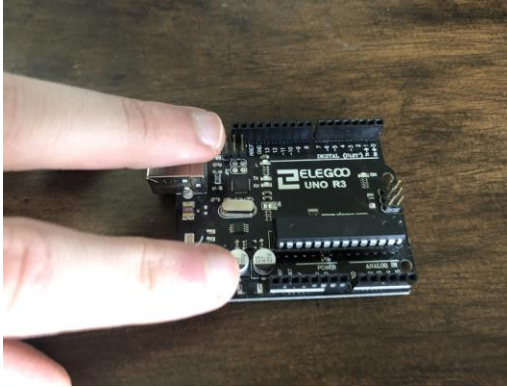
Figure 105 and 106, **Arduino Pin Sockets for LCD** (left)**, LCD Connection** (right)

The OSEPP LCD Module utilizes an Arduino Uno shield which fits its pins on the corresponding pin sockets on the Arduino board for testing. This is especially helpful for programming as there is no need to wire or solder any components to the Arduino chip itself.

This handy function for the OSEPP LCD Module is carried over to our PCB design to be fitted in similar pin sockets for interfacing with the Arduino Atmega328P, as well as any other board components that will be added to the PCB design.

In the fully integrated PCB design, this is further demonstrated to a working prototype to be produced in the final prototype physical design. It will work and look like the pictures below.
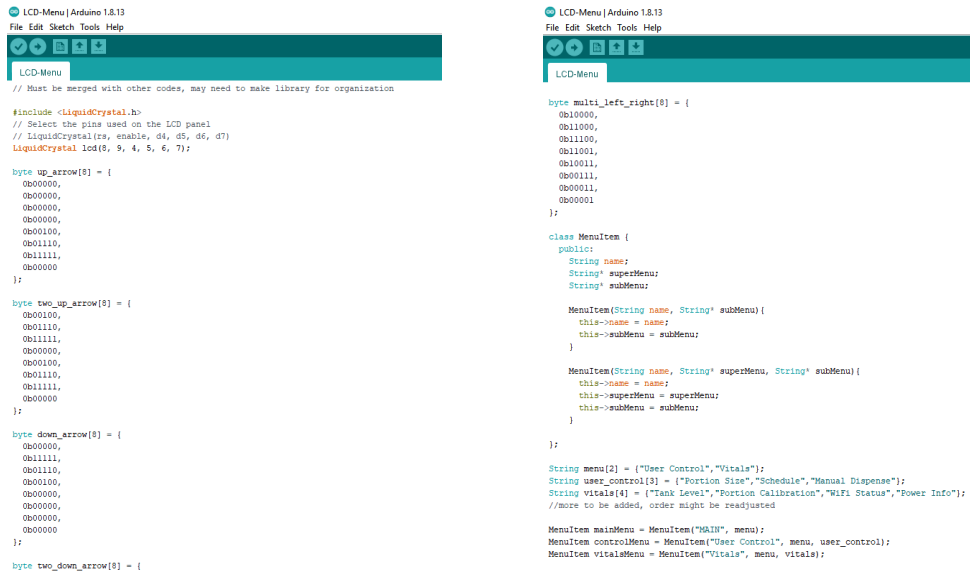
On startup, the code compiled to the Arduino Atmega328P MCU will display the startup screen then load into the menu. This will be further elaborated on in the LCD Software Implementation section below and will cover the basic code that is used to startup and run any processes that print out to the LCD screen.

The screen will have an interactive menu that the user will be able to cycle through and effectively read any sensor data along with any other information that may be desired. Also, the user will be able to use a few commands in the menu by simply following the simple but organized menu system.

Figure 107 and 108, **LCD Shield Assembled** (left), **LCD Startup** (right)

LCD Software Implementation

Figure 109 and 110, **LCD Custom Symbol Bytes** (left), **LCD MenuItem Class** (right)

In this section, the code for the LCD menu prototype is shown broken down into different sections. The code to the left shows the initial library that is an Arduino standard library for LCDs. The constructor shown will enable the corresponding pins on the Arduino to enable the specific purpose of those pins and what data they will process.

Also, shown is the byte array of custom symbols for the LCD segments. Included in this menu system are custom symbols for up, down, left, right and more symbols used for indicating more menu options along with others.

Each bit in each byte in the array corresponds to a pixel in each segment of the 16x2 display. Each segment is 5x8 pixels and can be viewed for ease by arranging it in the code shown here. To draw on the segment, 1 is entered to the byte at the specific bit and will display on the LCD when called. For the opposite, 0 does not display any pixel in the segment.

This section of the code to the right shows the class for a custom menu object that can be used for easy control of the menu item being displayed. Every menu item, except for the startup, is an object of the MenuItem class where it holds valuable components such as the name of the menu item, its parent menu and its child menu.

Each menu is a string array that holds the menu item's names that are its children. The prototyped code here shows just a view as they are continually adjusted to find a good flow for the menus.

Finally, the menu items are constructed and hold the values in the menu that can be easily accessed when the code below begins to become dense.

Some of the code shown in the snippet on the left is code that was given from the OSEPP LCD software section of the support site for this module.

Macros are set for buttons to later be used as names rather than numbers for organization within the code. The macros defined in the top section are used for the custom symbols which will be explained in the code snippet image below.

The remaining code for this code snippet image has variables and macros along with a function to read button input. The function shown reads the analog data from the analog-to-digital converter for pin 0 (subject to change) and has various if statements that follow a range for which button is pressed. That value is then returned as the macro value in an integer and can/will be used for traversing the menu along with any other interactive functions that the menu may utilize.

The code snippet image on the right is the start of the program within the setup function of the Arduino code. Functions of the LiquidCrystal library are called with the easily named macros defined above.

This full code is subject to change but will be represented inside of the loop() function which will house all the menu function calls and the button controls that correspond to those menu calls.

This menu will loop within the loop() function until a function is completed and the menu will leave the internal loop and start back at the top of the loop() function. This will ensure a continuing menu.

Figure 111 and 112,
Button Macros (left), **Menu Start** (right)


```
if (adc_key_in < 195)   return btnUP;
if (adc_key_in < 380)   return btnDOWN;
if (adc_key_in < 555)   return btnLEFT;
if (adc_key_in < 790)   return btnSELECT;
return btnNONE;  // when all others fail, return this...
}

// cursor control function here
```

```
MenuItem mainMenu = MenuItem("MAIN", menu);
MenuItem controlMenu = MenuItem("User Control", menu, user_control);
MenuItem vitalsMenu = MenuItem("Vitals", menu, vitals);

#define arrUp 0
#define dblArrUp 1
#define arrDown 2
#define dblArrDown 3
#define arrLeft 4
#define arrRight 5
#define arrLeftRight 6

// define some values used by the panel and buttons
int lcd_key     = 0;
int adc_key_in  = 0;
#define btnRIGHT 0
#define btnUP    1
#define btnDOWN  2
#define btnLEFT  3
#define btnSELECT 4
#define btnNONE  5

// read the buttons
int read_LCD_buttons()
{
  adc_key_in = analogRead(0);      // read the value from the sensor
  // my buttons when read are centered at these valies: 0, 144, 329, 504, 741
  // we add approx 50 to those values and check to see if we are close
  if (adc_key_in > 1000) return btnNONE; // We make this the 1st option for speed reasons since it will be the most likely result
  if (adc_key_in < 50)   return btnRIGHT;
  if (adc_key_in < 195)  return btnUP;
  if (adc_key_in < 380)  return btnDOWN;
  if (adc_key_in < 555)  return btnLEFT;
  if (adc_key_in < 790)  return btnSELECT;
  return btnNONE;  // when all others fail, return this...
}

// cursor control function here
```

```
void setup()
{
  lcd.createChar(arrUp,up_arrow);
  lcd.createChar(dblArrUp,two_up_arrow);
  lcd.createChar(arrDown,down_arrow);
  lcd.createChar(dblArrDown,two_down_arrow);
  lcd.createChar(arrLeft,one_left);
  lcd.createChar(arrRight,one_right);
  lcd.createChar(arrLeftRight,multi_left_right);

  lcd.begin(16, 2);                  // start the library
  lcd.setCursor(3,0);
  lcd.print("Welcome to");
  lcd.setCursor(1,1);
  lcd.print("Food Dispenser");
  delay(5000);
  lcd.clear();
  delay(1000);
}
```

5.3 Power supply

Voltage Regulator Circuit

Voltage regulation for this design is very simple since we only need two levels of voltage regulation, 5v and 3.3v. These voltages will be supplied to different components of the design. As we mentioned earlier in the research section of this document, linear regulators are the best choice for the voltage regulation needed for our pet feeder.
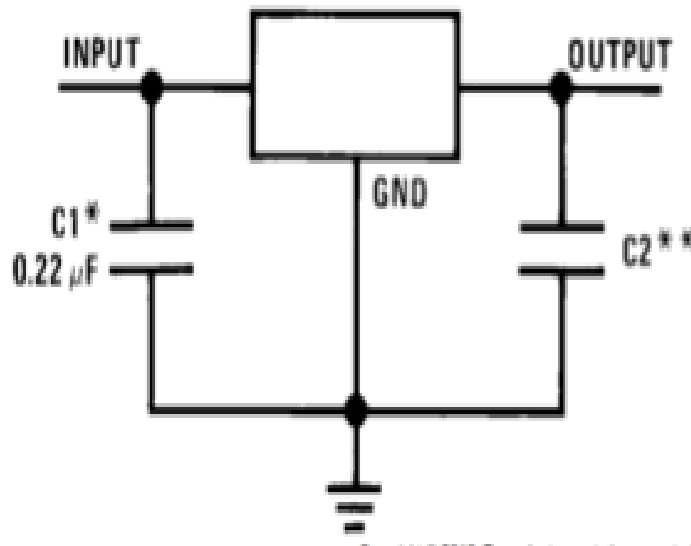

Figure 113, **Typical application of a fixed output voltage regulator**

When we selected linear regulators, we only needed to meet a few specifications. For our design, we need a transformer to step down the wall outlet voltage to a range that would meet the regulator's input specifications. These specifications are typically between 12v and 40v as input voltage to the regulator which will in turn drop this input voltage to an output of 5v and 3.3v and 1.5A as the output current. Our final choice for the voltage regulator is to use linear regulators that will meet these requirements. We ended up choosing the Texas instruments LM7805 fixed output voltage to supply the 5v to the main PCB. We also ended up selecting the LD1117 to supply the 3.3v required to power the wi-fi module. These linear regulators are rated at 1.5A and 1A which is sufficient to accomplish what we need to power the pet feeder.

Figure 114, **LM340 / LM7805 Electrical Characteristics**

PARAMETER		TEST CONDITIONS		MIN	TYP	MAX	UNIT
V_O	Output voltage	$T_J = 25°C$, 5 mA $\leq I_O \leq$ 1 A		4.8	5	5.2	V
		$P_D \leq$ 15 W, 5 mA $\leq I_O \leq$ 1 A 7.5 V $\leq V_{IN} \leq$ 20 V		4.75		5.25	V
ΔV_O	Line regulation	I_O = 500 mA	$T_J = 25°C$ 7V $\leq V_{IN} \leq$ 25V		3	50	mV
			Over temperature 8V $\leq V_{IN} \leq$ 20V			50	mV
		$I_O \leq$ 1 A	$T_J = 25°C$ 7.5V $\leq V_{IN} \leq$ 20V			50	mV
			Over temperature 8V $\leq V_{IN} \leq$ 12V			25	mV
ΔV_O	Load regulation	$T_J = 25°C$	5 mA $\leq I_O \leq$ 1.5 A		10	50	mV
			250 mA $\leq I_O \leq$ 750 mA			25	mV
		Over temperature, 5 mA $\leq I_O \leq$ 1 A				50	mV
I_Q	Quiescent current	$I_O \leq$ 1 A	$T_J = 25°C$			8	mA
			Over temperature			8.5	mA
ΔI_Q	Quiescent current change	0°C $\leq T_J \leq$ 125°C, 5 mA $\leq I_O \leq$ 1 A			0.5		mA
		7 V $\leq V_{IN} \leq$ 20 V	$T_J = 25°C$, $I_O \leq$ 1 A			1	mA
			Over temperature, $I_O \leq$ 500 mA			1	mA
V_N	Output noise voltage	$T_A = 25°C$, 10 Hz $\leq f \leq$ 100 kHz			40		μV
$\dfrac{\Delta V_{IN}}{\Delta V_{OUT}}$	Ripple rejection	f = 120 Hz 8 V $\leq V_{IN} \leq$ 18 V	$T_J = 25°C$, $I_O \leq$ 1 A	62	80		dB
			Over temperature, $I_O \leq$ 500 mA	62			dB
R_O	Dropout voltage	$T_J = 25°C$, I_O = 1 A			2		V
	Output resistance	f = 1 kHz			8		mΩ
	Short-circuit current	$T_J = 25°C$			2.1		A
	Peak output current	$T_J = 25°C$			2.4		A
	Average TC of V_{OUT}	Over temperature, I_O = 5 mA			−0.6		mV/°C
V_{IN}	Input voltage required to maintain line regulation	$T_J = 25°C$, $I_O \leq$ 1 A		7.5			V

After deciding on the right voltage regulator for the power supply, we proceeded to build a prototype that would allow our group to start testing. We used basic components to build the two voltage levels required for our project. We built the circuit on a breadboard using a combination of resistors, diodes, capacitors, and voltage regulators. Working with the breadboard was not complicated since we were familiar with building these circuits during the lab experiments. Therefore, from experience these components are first checked with a multimeter to assure they have the correct values that we requested.
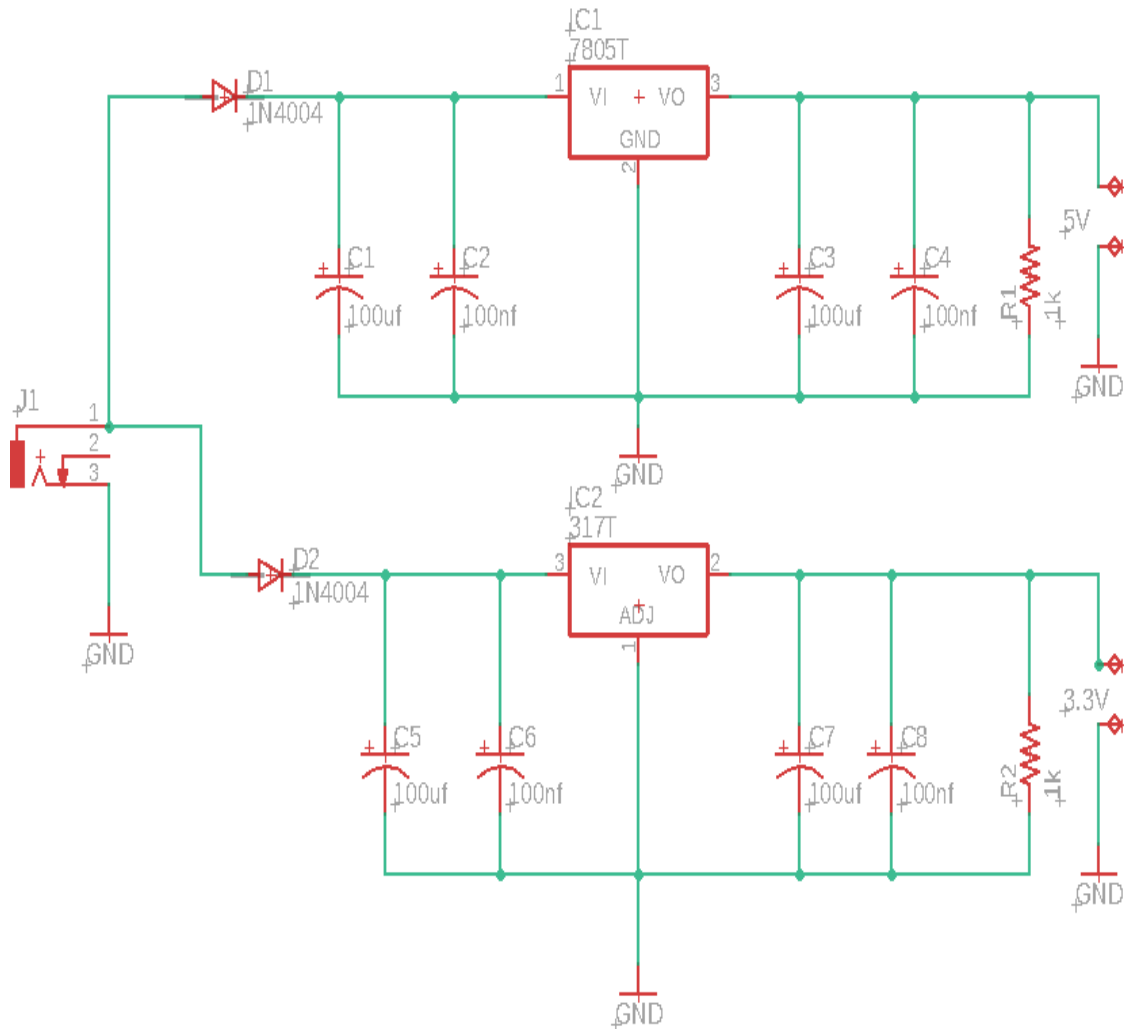
Despite the fact these parts were purchased and arrived in sealed packages, one cannot always assume that they have the right specifications. We checked each part to make sure it has the desired value that we need to use for our circuit. We learned to always take this step first before placing the parts into the breadboard to avoid using a defective component and having to start over after having the circuit already setup which is time consuming. This is very critical also when it comes to troubleshooting, when issues arise or when the desired results are not obtained. If each individual part was checked prior to being placed in the breadboard but the final output does not match what was expected then one can safely assume the circuit design is the issue. After testing each single part, we continued to built our circuit prototype on the breadboard as shown below:

Figure 116, breadboard testing of 5V
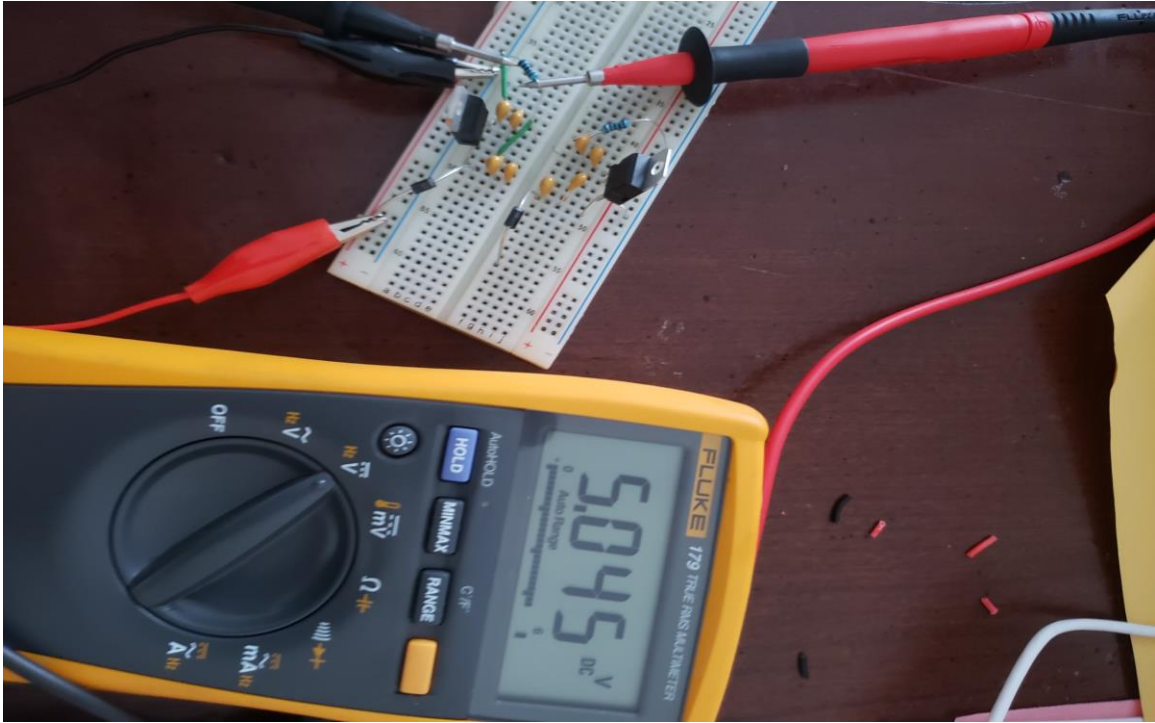
Figure 117, **breadboard testing of 3.3V**
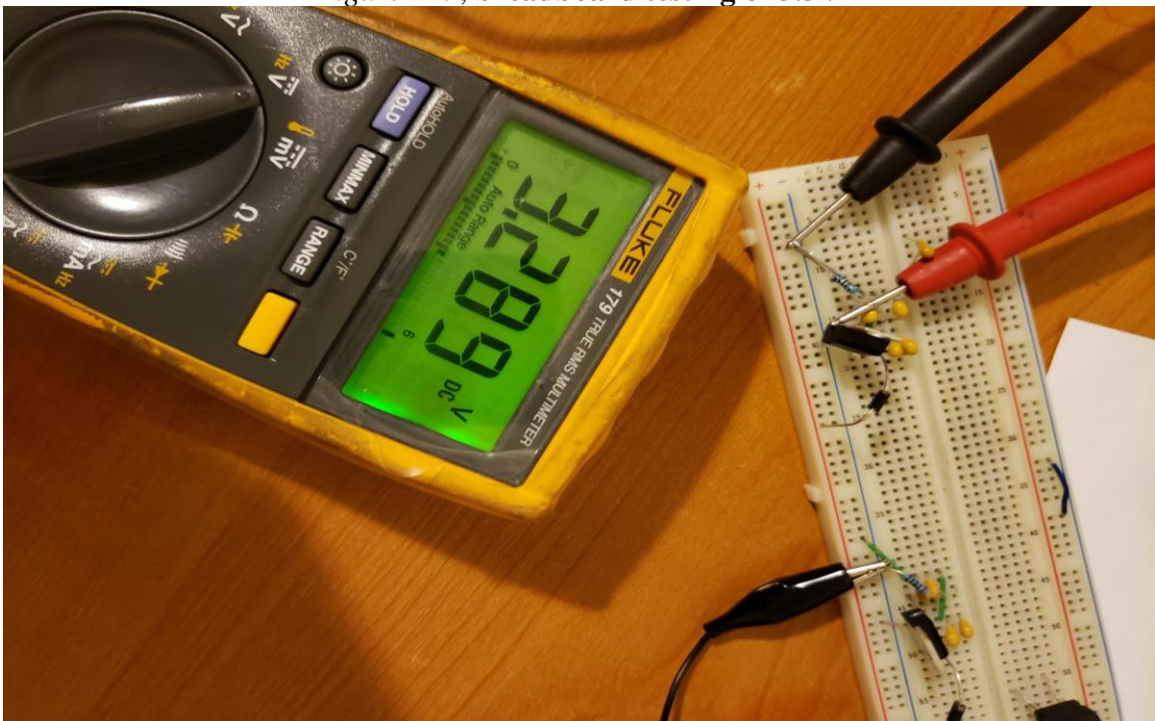
We can see that the results we obtained from our prototype circuit are what we actually expected. Both the 5v and the 3.3v circuits are meeting our expectations. This is a crucial step in the design because we have to assure proper voltage supplies to the rest of the

components. However, this type of breadboard prototype is not allowed in senior design, we will have to build a professional printed circuit board that will contain all these components in a professionally designed manner.

The next step is to order the parts used in the prototype and start building the power supply. We started by designing a printed circuit board using one of the numerous platforms available for this purpose. Since we are already familiar with Eagle software that we used in junior design class, we used it to build our schematic and design the PCB for the power supply. The advantage of using Eagle is that it offers a large content library of components from different manufacturers, it also offers a price quote for the PCB manufacturing as well as a bill of material. In addition to these advantages, it also provides a schematic sheck option which allows the designer to let the software check for any error in the design. If aunty errors were detected, Eagle pinpoints these errors to make it easy on the designer to go back and make any necessary changes to correct those errors.The final schematic design that we ended up with is shown below:

Figure 118, **Eagle PCB design**
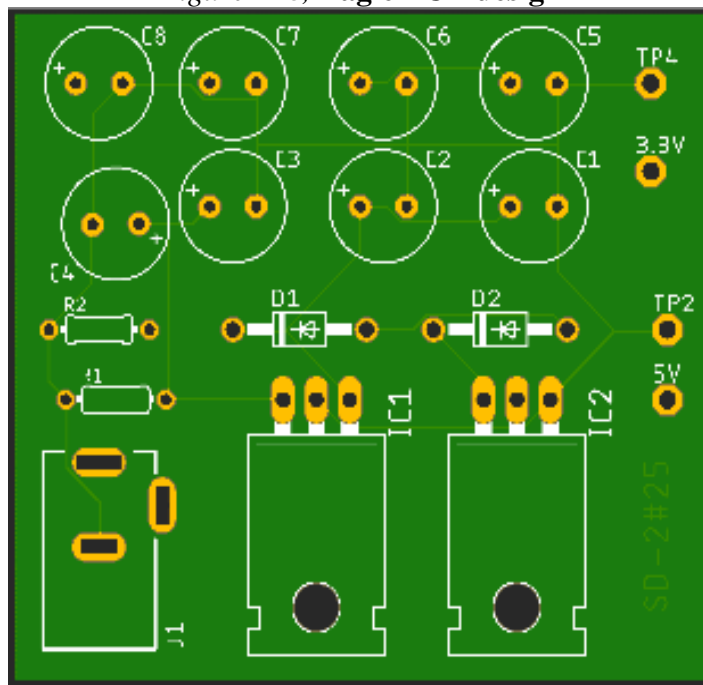
When designing the power supply, we made multiple attempts at the schematic design. After completing each design, we run a quote for the price of the PCB as well the BOM and the lead time from different manufacturers or part suppliers. We found that there are long delays for certain parts, this puts more constraints on our design since we have certain deadlines to meet. The delays for the parts manufacturing delivery is mostly due to the pandemic which forced some manufacturers to shut down. In addition to that, some countries are still having forced lockdowns which makes the logistics more difficult. This is the main reason why we had to make multiple design attempts. Each time, we have to modify certain components; we have to remove the parts that have long delays or low

availability and replace them with parts that are readily available and with less delays. The final PCB layout with the components placement is shown below:

Figure 119, **Eagle PCB layout**
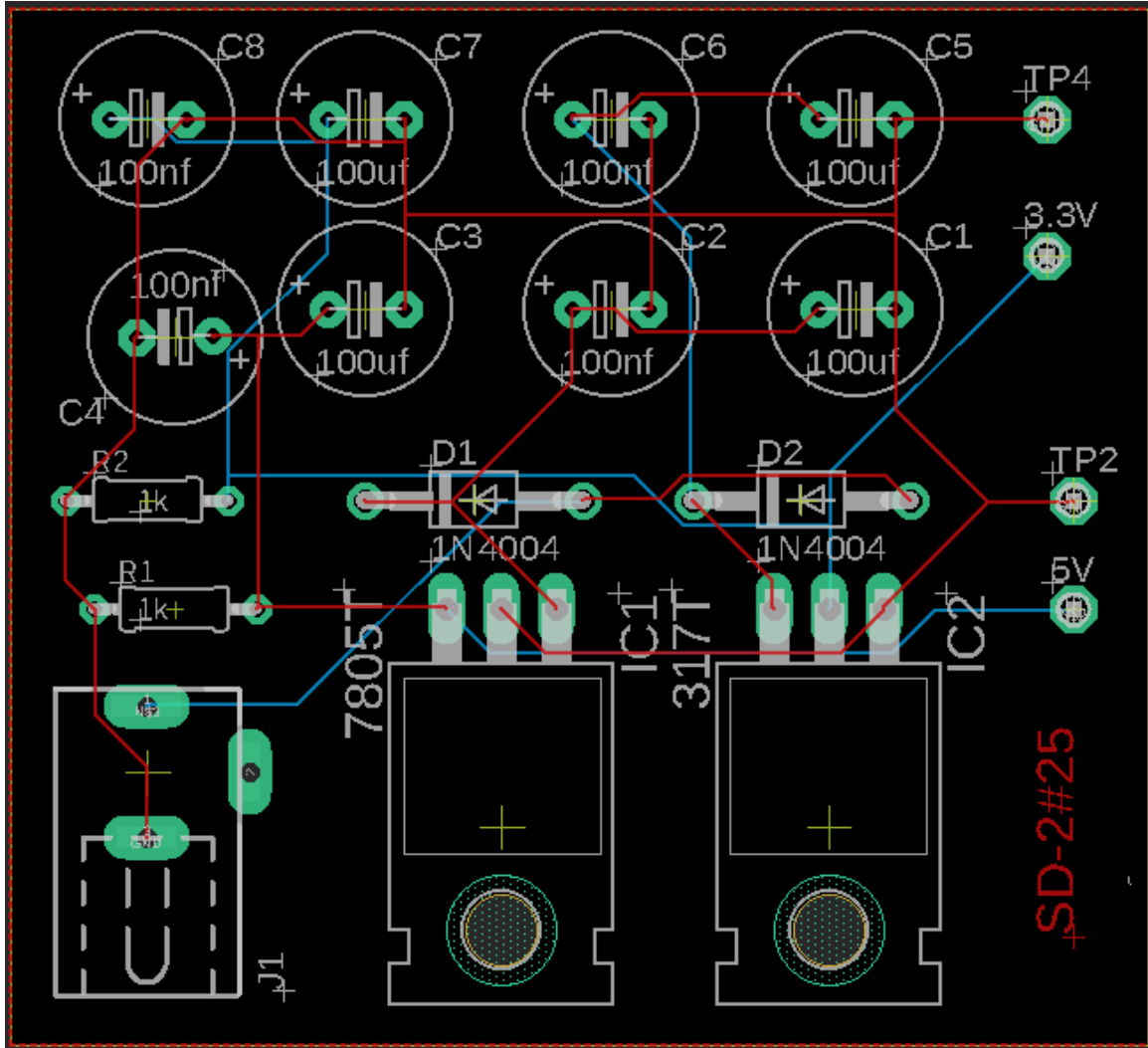
5.4 Sensors

HC-SR04 Ultrasonic Testing

This is a test for the HC-SR04 Ultrasonic sensor to see if it can accurately detect objects. Below is a diagram of what the circuit for the HC-SR04 sensor should be. There are 4 pins for the sensor, 2 of them are for power: Vcc and GND. The other 2 is for operation: Trigger and Echo. HC-SR04 sensor requires 5V of power, which is supplied by the Arduino Uno's chip: ATMEGA328P. The Trigger and Echo pins can be connected to any digital pins. Due to most of the Arduino's digital pins functioning the same, any of the digital pins can be

used. With the requirements of other modules, the HC-SR04 is planned to use digital pins D3 and D4 for Trigger and Echo respectively.

Figure 120, **HC-SR04 Ultrasonic Sensor circuit mapping diagram**
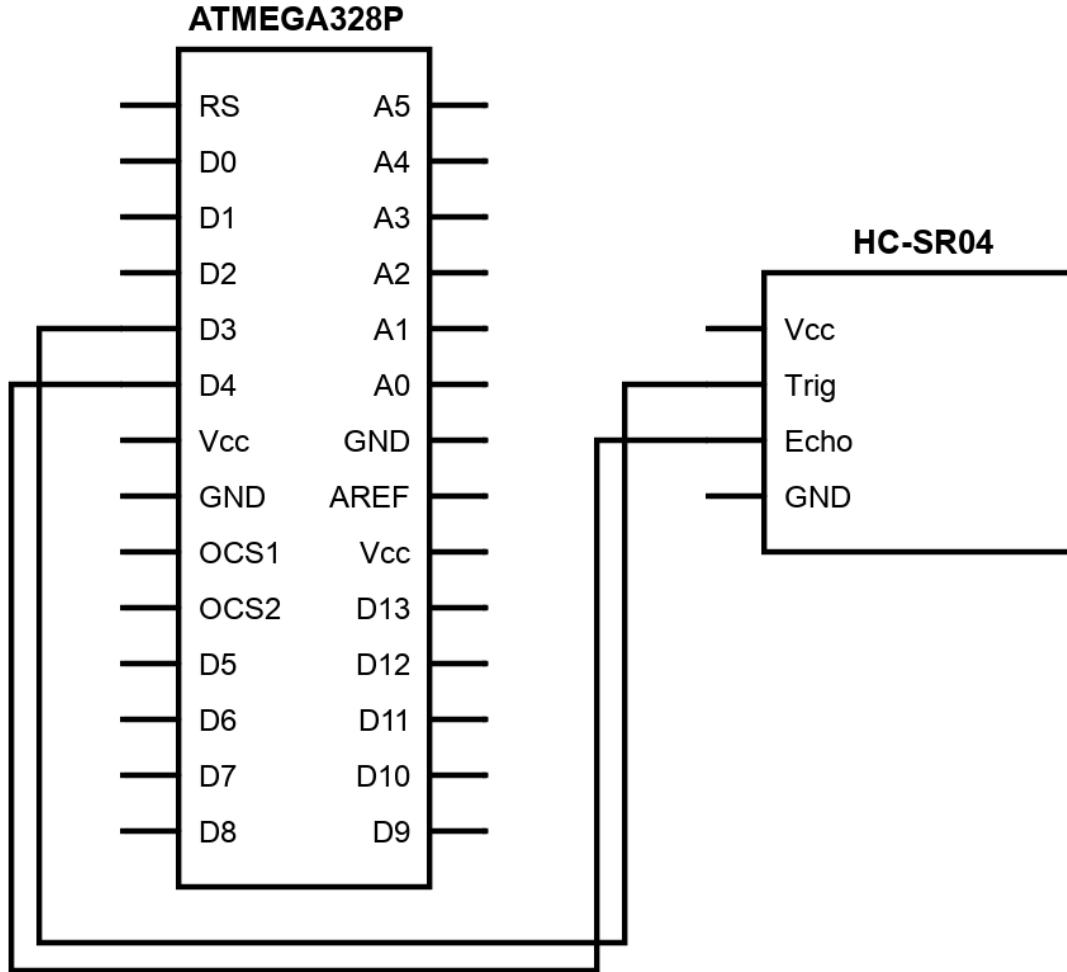
Here is the actual circuit for the ultrasonic sensor using the Arduino Uno. This circuit design is for testing and demonstration purposes only. It is not intended to be used for the final design and is subject to change. The purpose of this circuit is only to make sure all components and code work properly so they may be used in the final build.

Figure 121, **HC-SR04 Ultrasonic Sensor testing circuit**
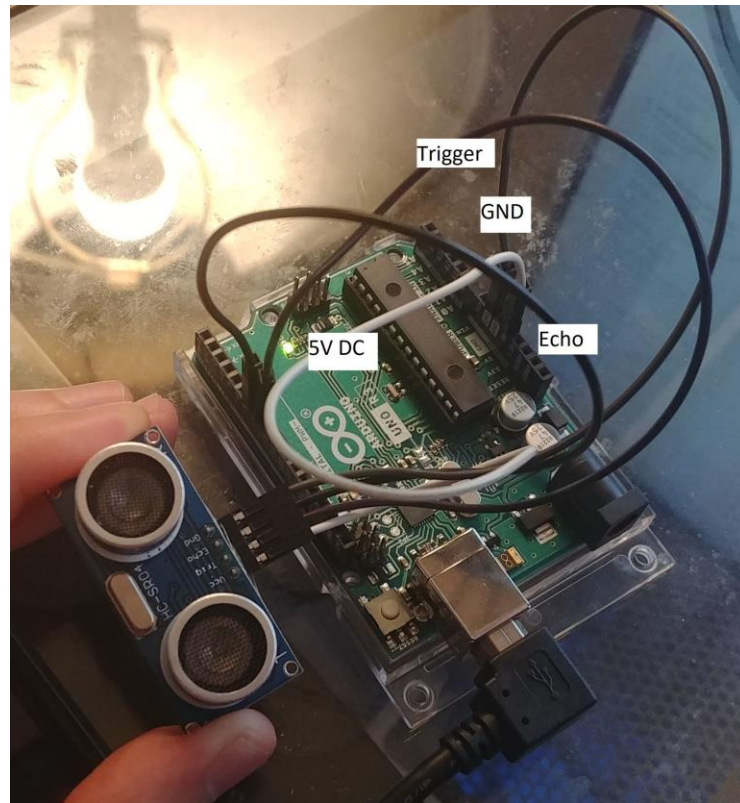
Here is a test of the sensor and its results. The code rechecks the sensor every .1sec and gets the distance results from it. The results are displayed in inches and centimeters.

For this test, I used my hand as the object. I kept my hand in the same place and recorded the data for 10 seconds to see if any noise might mess up the sensor. Surprisingly, the sensor was constant in its measurements, only changing by 1 or 2 cm if my hand was shaking.

Figure 122, **HC-SR04 Ultrasonic Sensor detecting an object**
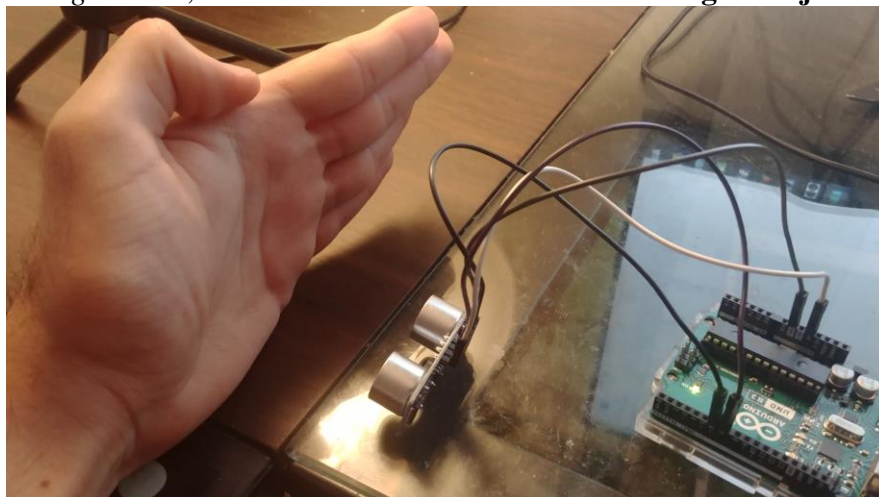
My hand was placed on the deck as a wall for the ultrasonic waves. The HC-SR04 was propped up on its board so it wouldn't move. The ultrasonic waves would bounce off my hand and rebound back to the sensor. The time it takes to return back would be the distance. Calculations for the distance from time is explained in the code.

Figure 123, **HC-SR04 Ultrasonic Sensor Distance Results**
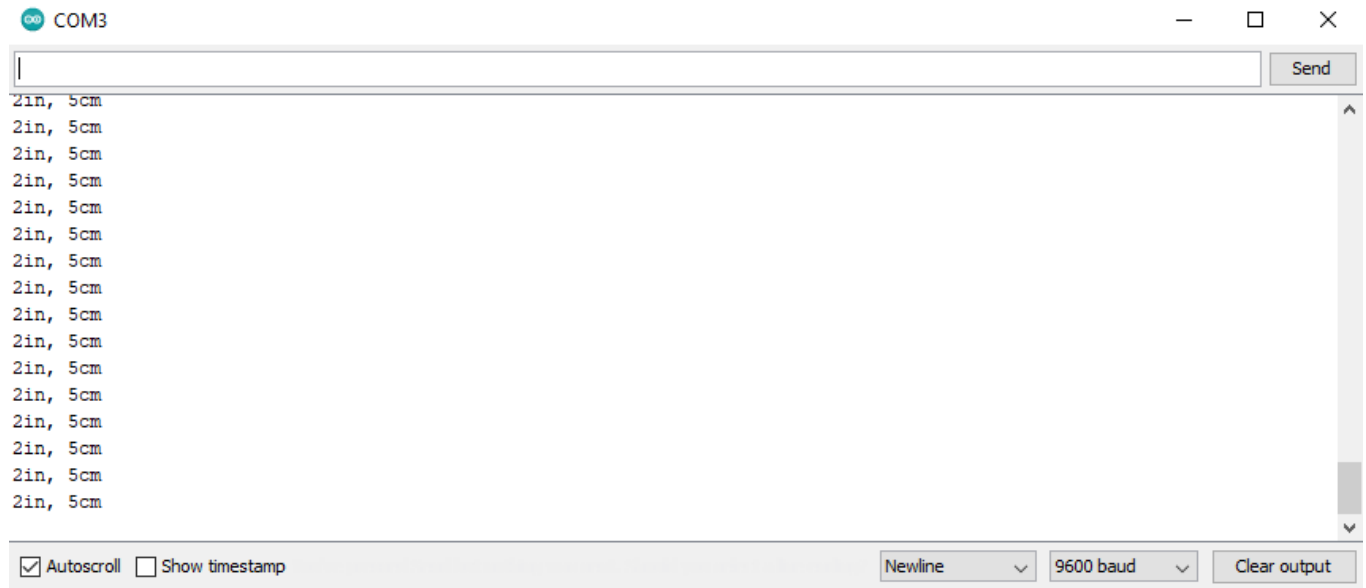
As shown in the figure above, the reading would always be constant 2 inches, 5centimeters. The distance from the sensor is reliable and accurate. All pins worked properly and were receiving enough power. The requirements this sensor needs: being able to detect an object is met. The sensor was able to easily identify objects over 50 cm away, which is more than enough for its purpose in the food container.

While the testing above was not performed with a breadboard, it can easily be placed on one for the final design. Overall, this part can be used properly for further testing and the code can be manipulated for the final product.

Figure 124, **HC-SR04 Ultrasonic Sensor Test Code**

```
const int pingPin = 7; // Trigger Pin of Ultrasonic Sensor
const int echoPin = 6; // Echo Pin of Ultrasonic Sensor

void setup() {
   Serial.begin(9600); // Starting Serial Terminal
}

void loop() {
   long duration, inches, cm;
   pinMode(pingPin, OUTPUT);
   digitalWrite(pingPin, LOW);
   delayMicroseconds(2);
   digitalWrite(pingPin, HIGH);
   delayMicroseconds(10);
   digitalWrite(pingPin, LOW);
   pinMode(echoPin, INPUT);
   duration = pulseIn(echoPin, HIGH);
   inches = microsecondsToInches(duration);
   cm = microsecondsToCentimeters(duration);
   Serial.print(inches); //display the inches
   Serial.print("in, "); //display unit
   Serial.print(cm); // display the cm
   Serial.print("cm"); // display unit
   Serial.println();
   delay(100); //delay of 100 ms or .1s
}

long microsecondsToInches(long microseconds) {
  // According to Parallax's datasheet for the PING, there are
  // 73.746 microseconds per inch (i.e. sound travels at 1130 feet per
  // second).
  // The sound has to go back and forth, so we divide by 2.
   return microseconds / 74 / 2;
}

long microsecondsToCentimeters(long microseconds) {
  // Sound travels at approximately 340 meters per second.
  // This corresponds to about 29.412µs (microseconds) per centimeter.
  // The sound has to go back and forth, so we divide by 2.
   return microseconds / 29 / 2;
}
```

HX771 Amplifier with Weight Load Cell Testing

This is a test for the HX771 amplifier and the Weight Load Cell to see how it measures weight. The load cell includes 4 wires/pins all with different colors: red for excitation + (Vcc), black for excitation - (GND), green for output + and white for output -. Each respective colored wire goes to the proper HX711 slot. The HX711 connects to the ATMEGA328P chip very similar to the HC-SR04. Vcc and GND are connected to each other and DT and SCK are connected to any of the digital pins. With the requirements of the other modules, the HX711 amplifier is planned to connect to digital pin 5 and 6.

Figure 125, **HX711 with Weight Load Cell to Arduino**
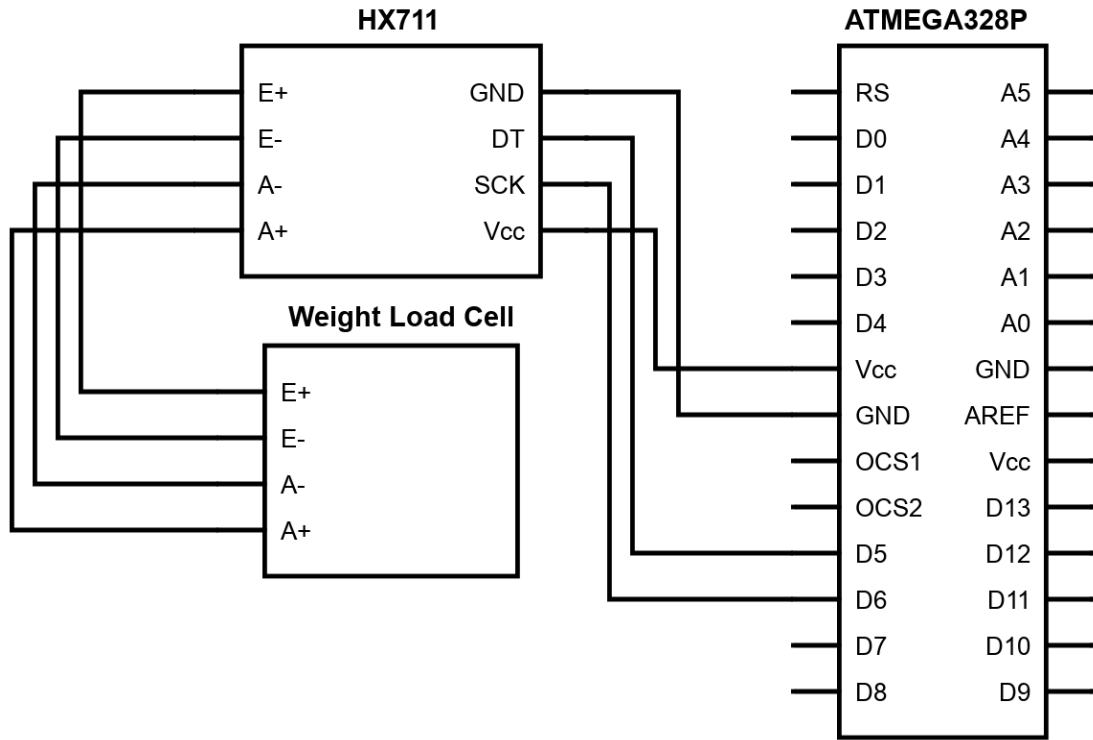
One thing that is different from the HC-SR04 ultrasonic sensor is that using the load cell requires calibration before use. Due to many different scales and manufacturers, there are many different calibration factors. One of the many ways to calibrate weights is using the phone test. By weighing the phone and manually inputting its weight, you can easily calibrate the scales. Some of the figures below are temporary sample figures and not my own.

Figure 126, **Phone being weighed tested**
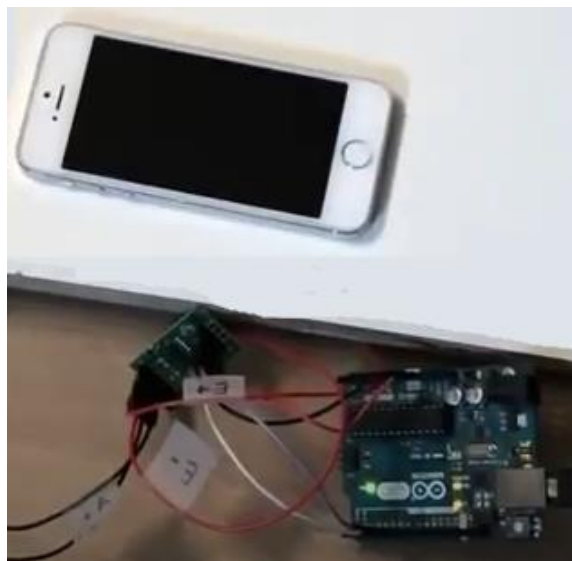
The phone is being weighted on the sensor. An IPhone 5 typically weighs about 113 grams, so we will input that for our calibration.

Figure 127, **HX711 with Weight Load Cell to Arduino**

```
113

Starting...
Startup + tare is complete
***

Start calibration:
It is assumed that the mcu was started with no load applied to the lc
Now, place your known mass on the loadcell,
then send the weight of this mass (i.e. 100.0) from serial monitor.
```

Figure 128, **Load Cell Calibration Code**

```cpp
#include "HX711.h"

#define LOADCELL_DOUT_PIN  3
#define LOADCELL_SCK_PIN  2

HX711 scale;

float calibration_factor = -7050; //-7050 worked for my 440lb max scale setup

void setup() {
  Serial.begin(9600);
  Serial.println("HX711 calibration sketch");
  Serial.println("Remove all weight from scale");
  Serial.println("After readings begin, place known weight on scale");
  Serial.println("Press + or a to increase calibration factor");
  Serial.println("Press - or z to decrease calibration factor");

  scale.begin(LOADCELL_DOUT_PIN, LOADCELL_SCK_PIN);
  scale.set_scale();
  scale.tare(); //Reset the scale to 0

  long zero_factor = scale.read_average(); //Get a baseline reading
  Serial.print("Zero factor: "); //This can be used to remove the need to tare the scale. Useful in permanent scale projects.
  Serial.println(zero_factor);
}

void loop() {

  scale.set_scale(calibration_factor); //Adjust to this calibration factor
  Serial.print("Reading: ");
  Serial.print(scale.get_units(), 1);
  Serial.print(" lbs"); //Change this to kg and re-adjust the calibration factor if you follow SI units
  Serial.print(" calibration_factor: ");
  Serial.print(calibration_factor);
  Serial.println();

  if(Serial.available())
  {
    char temp = Serial.read();
    if(temp == '+' || temp == 'a')
      calibration_factor += 10;
    else if(temp == '-' || temp == 'z')
      calibration_factor -= 10;
  }
}
```

After the calibration is finished, the scale can be run for testing purposes. Here is a test of the sensor and its results. The code rechecks the sensor every .1sec and gets the weight results from it. These are the results for a 200g weight and the results are displayed in grams.

Figure 129, **Load Cell Weight Results**

*Figure 130*, **Load Cell Calibration Code**

```
#include "HX711.h"
#define DOUT  4
#define CLK  5
HX711 scale(DOUT, CLK);

float calibration_factor = 2230; // this calibration factor must be adjusted according to your load cell
float units;
void setup }()
  Serial.begin(9600);
  Serial.println("HX711 calibration sketch");
  Serial.println("Remove all weight from scale");
  Serial.println("After readings begin, place known weight on scale");
  Serial.println("Press + or a to increase calibration factor");
  Serial.println("Press - or z to decrease calibration factor");

   scale.set_scale(calibration_factor); //Adjust to this calibration factor
  scale.tare();  //Reset the scale to 0

  long zero_factor = scale.read_average(); //Get a baseline reading
  Serial.print("Zero factor: "); //This can be used to remove the need to tare the scale. Useful in permanent scale projects.
  Serial.println(zero_factor);
 {

void loop}()
  float average = 0;
  int count = 0;
  Serial.print("Reading");
  units = scale.get_units(), 5;
  if (units < 0)
 }
    units = 0.00;
  {
  Serial.print("one reading: ");
  Serial.print(units);
  count++;
  average = units / count;
  Serial.print("         | average:    ");
  Serial.print(average);
  Serial.println();
```

From this simple test, we can see that the scale is not exact, but pretty close. There is a little variation but that will need to be adjusted for the final product. The final design will not have such a big surface area so results will vary. However for a simple test of the bathroom scale load cell, we can see that the weight is very close to the actual. All components and connections work properly on their own.

5.5 Wifi and Mobile

Circuit Design

<center>*Figure 131*, **Connection Circuit ESP8266 to Arduino Uno**</center>
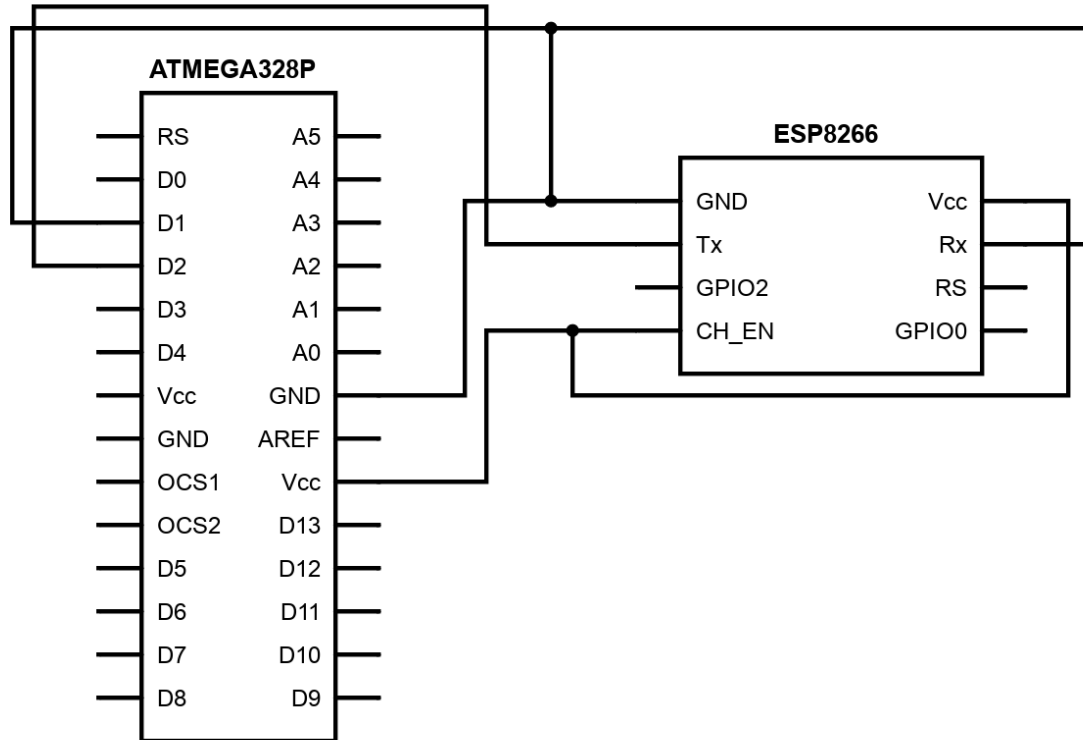
Here we have our design of how the ESP8266 wifi module will connect to the Arduino Uno's ATMEGA328P chip. We can start off by knowing that the three pins not used on the ESP8266 are the GPIO2 pin, GPIO0 pin, and the RS pin. GPIO stands for the general input/output pin and RS stands for the reset pin. We see that the Tx pin of the ESP8266 is connected to the digital pin number 2 of the Arduino Uno. The channel enable and Vcc of the ESP8266 is connected to the 3.3V Vcc of the Arduino uno this is how the wifi module is turned on and is able to connect to the internet. Next we can see that the ground on both chips are connected together to make a full circuit. Finally we have the Rx that is connected to the ground and digital pin number 1.

This is a good representation but is missing some of the resistors used to regulate the voltage between the two boards. There should be a 10 kohm resistor separating the Vcc from the Arduino and the Vcc and channel enable on the ESP8266. There are also two 1 kohm resistors missing, one separating the Rx and the digital pin number 1 and the Rx connected to ground. These are important because they are used to regulate the voltages so they do not damage any of the pins on both of the boards.

This image is made to represent how each board is connected to one another not to show how the voltage is regulated. The wiring on this image shows what pins the wifi module is using so other devices can not use the same ones.

Breadboard Testing
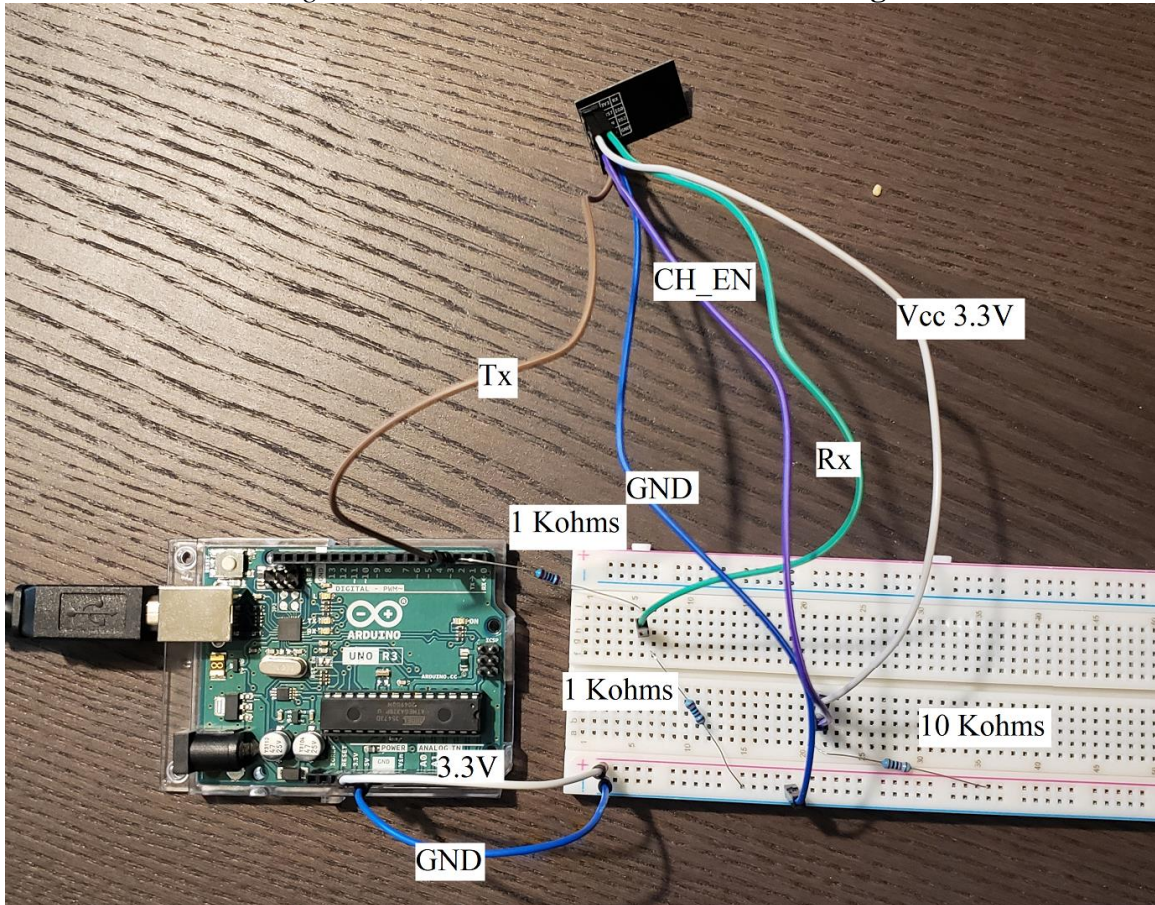
Here we have the ESP8266 connected to the ArduinoUno with the help of the breadboard. We were able to get the Arduino and the wifi module to connect. We can see how each of the pins from the wifi module connects to the Arduino. We can also see that there are some resistors present to create a voltage divider to lower the incoming voltage from the Arduino because it has an operation voltage of 5V while the ESP8266 can only handle 3.3V.

After getting the ESP8266 physically connected to the Arduino MCU testing it was a little bit more challenging. Since the Arduino did not come with any power supply or wire to program, it took a lot of time to find the right tools needed to be able to upload or even turn on the board. Once all the right tools were able to be acquired we can then begin the testing.

To test the wifi module we had to upload some code, that can be found below, so it can prepare the wifi module for connection and communication. we were able to do that, but we do not know if the code was completely correct or if my schematic was off, because we were not able to connect to it using the TCP client. This is something we need to look more into to make sure everything is working correctly or if there may be a broken part.

Flow Chart

Figure 133, **Mobile Application Control Flow chart**
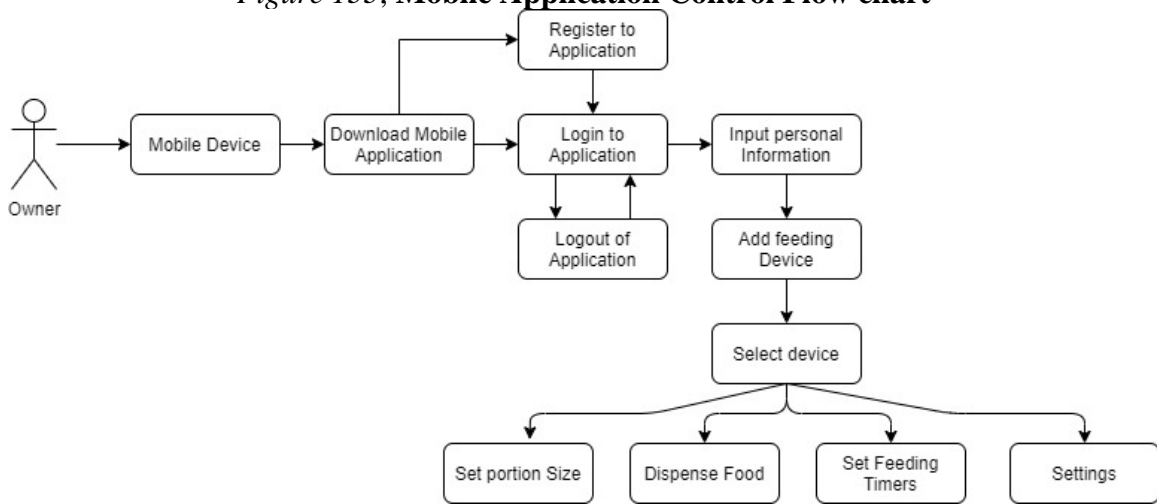
This image is a flow chart of how the mobile application should work for all of its users. We can see it first starts with the user or client. The user and/or client then would need to use their mobile device to download the mobile that was created for the automatic pet feeder. Once the application is downloaded the user will be directed to the login page and be asked to login, if the user is new and does not have an account, there will be an option to register an account. Once an account is registered the user can then login and start using the mobile app.

Once in their mobile application we see that they can input more personal information into the account, for themselves or for their pets. When the account is all set up and personalized they can then proceed to add their automatic pet feeder. We can then move on to selecting the device and controlling it from the mobile application.

We can see some of the main control features of the mobile application is Setting portion sizes such as snack, small, medium, and large. This is gonna be based on the recommended serving values for the pets from the veterinarians.

Another key control feature is dispensing food manually here the user can select and dispense the meal serving right away to their pets. This is a great method if users are not comfortable with setting timers and want to do it manually to make sure their pets are getting fed on irregular schedules.

We then have the Set timer feature that allows users to set timers on their device so there can be a regular eating schedule for their pets. This can be very useful for people who just tend to forget to put out food or are always on the go and do not necessarily have the time to put it out sometimes. Finally we have the setting page where users can see and alter any of their personal information. There is also a logout function that allows the user to log out of the app entirely but that will not be used too often.

Software

Figure 134, **Sample Code Snippet of wifi Setup**

```
#include <SoftwareSerial.h>

SoftwareSerial wifiSerial(2, 3);        // RX, TX for ESP8266

bool DEBUG = true;   //show more logs
int responseTime = 10; //communication timeout

void setup()
{
  pinMode(13,OUTPUT);   //set build in led as output
  // Open serial communications and wait for port to open esp8266:
  Serial.begin(115200);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for Leonardo only
  }
  wifiSerial.begin(115200);
  while (!wifiSerial) {
    ; // wait for serial port to connect. Needed for Leonardo only
  }
  sendToWifi("AT+CWMODE=2",responseTime,DEBUG); // configure as access point
  sendToWifi("AT+CIFSR",responseTime,DEBUG); // get ip address
  sendToWifi("AT+CIPMUX=1",responseTime,DEBUG); // configure for multiple connections
  sendToWifi("AT+CIPSERVER=1,80",responseTime,DEBUG); // turn on server on port 80

  sendToUno("Wifi connection is running!",responseTime,DEBUG);


}
```
Here we have the code to set up the wifi module with the Arduino uno this will allow the module to communicate and send data back and forth with the Arduino allowing it to also be interacted with through commands via the internet. First we can see that there is a library called SoftwareSerial.h that has to be imported. We then have to set the digital pins for the Rx and Tx of the ESP8266 for data communication. This here is important so we know where the data is being sent to and from for both the wifi module and MCU. We then have to allow debugging to catch errors if a message cannot be sent or read properly then we can send a message notifying the user something has gone wrong. We will also include a response time to make sure that a command does not stall or get stuck or lost. If this were to happen the time out will stop the current command and send an error notifying the user the command was not successfully executed.

Once all those are set up we can then go over to the setup function this is where we can start to configure the wifi module such as opening a serial port and waiting to connect to the module. Here we will also configure the access point and obtain the ip address to know where to send information to. The port will also be set to 80 because that is usually the most used port for wifi communication unless you want a more secure server which will be 443.

The difference between port 80 and 443 is that all things sent in 443 are encrypted and the ones sent from 80 are not.When all that is done you are connected and ready to go.

```
void loop()
{
  if(Serial.available()>0){
     String message = readSerialMessage();
    if(find(message,"debugEsp8266:")){
       String result = sendToWifi(message.substring(13,message.length()),responseTime,DEBUG);
       if(find(result,"OK"))
         sendData("\nOK");
       else
         sendData("\nEr");
    }
  }
  if(wifiSerial.available()>0){

     String message = readWifiSerialMessage();

    if(find(message,"esp8266:")){
       String result = sendToWifi(message.substring(8,message.length()),responseTime,DEBUG);
       if(find(result,"OK"))
         sendData("\n"+result);
       else
         sendData("\nErrRead");                  //At command ERROR CODE for Failed Executing statement
    }else
    if(find(message,"DISPENSE")){  //receives DISPENSE from wifi
       //turn on Motor to Release food
       digitalWrite(13,HIGH);
    }
    else{
       sendData("\nErrRead");                    //Command ERROR CODE for UNABLE TO READ
    }
  }
  delay(responseTime);
}
```

Here is the loop function that runs indefinitely waiting for commands from the mobile application. In this function we will have all of our functions such as set timer, dispense food, and portion control. This portion of the code will wait for a command either a custom one such as the ones I just mentioned or built in AT commands. We see here that the serial port must be available before we can send a command, this is to make sure that only one command is being sent at a time, if another command is still in process we would not want to send anything that would disturb what is already doing on. If there is already a command in process there would just be a slight delay before they can try to send another command.

While having an open channel is something we are looking for, another thing we need to look out for are unknown commands. Every command must match exactly to the custom ones or the AT commands otherwise there will be an error and cause the requested command to terminate. We have to have these checkers for unknown commands because it might cause our application or connection to the wifi device to end prematurely.

Finally we would also like to be able to return something after each command to know if it was a success or failure. This can be helpful for users to get notified if what they were asking for was actually done. If every function returned a 1 for success and 0 for failure then we can then use that information to notify the user.

Interface

*Figure 136,* **Custom Drawn Login page**
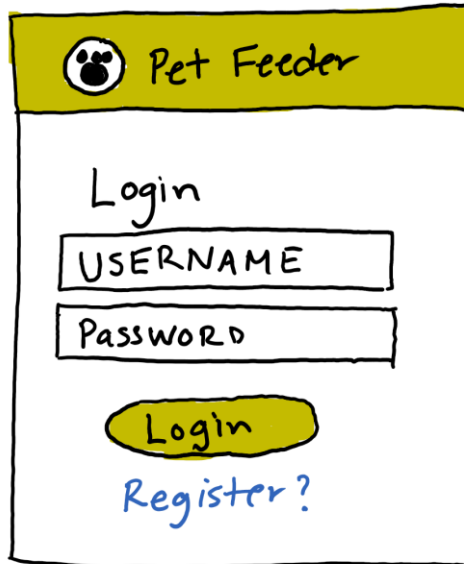
The image represents the login page. This will be the page you will be directed to when you first open the mobile application. We can see that if you already have an account you will just have to enter your username and password and login if not there is a register text button to redirect you to the register page.

*Figure 137,* **Custom Drawn Register Page**
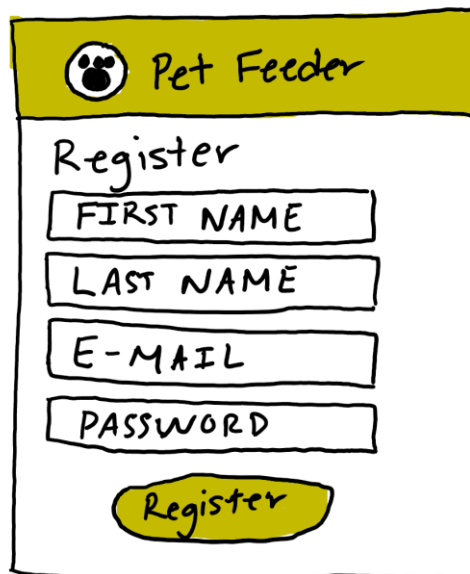
If you need to register this is what the page will most likely look like. Here to create an account you will need to provide a first and last name, with an email and password. Once all information is input correctly the user will have to just click the register button where they will be added as a user and get redirected to login to login to the application.

Figure 138, **Custom Drawn Home Page**
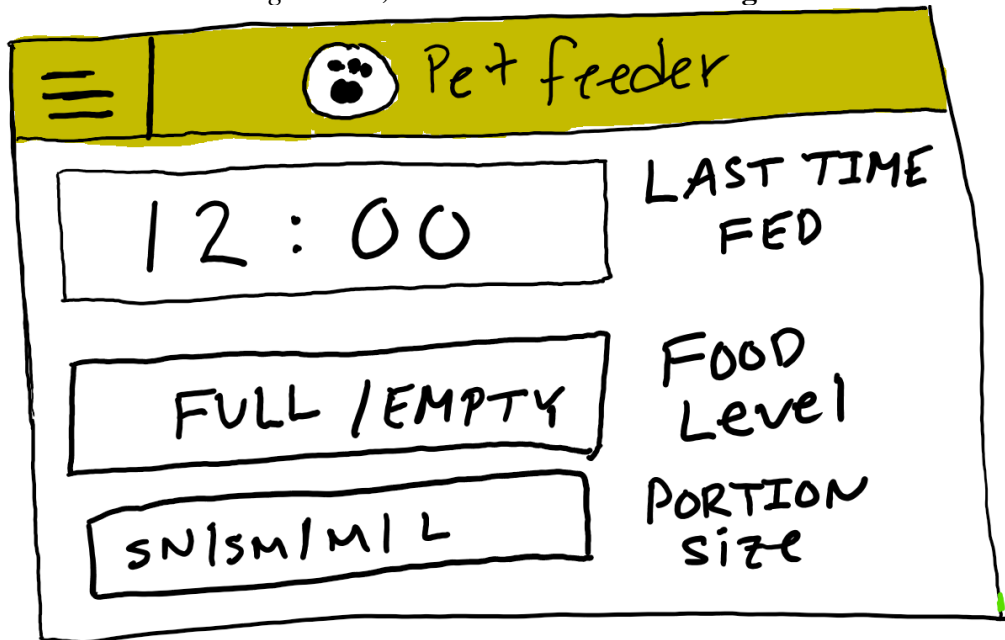
Above is an image of the home screen that will automatically load once the user has logged into the mobile application. Here there will be basic information such as when your pet was last fed, the food level, and the portion size that is currently selected. You can also see there is the name of the product and the logo on the top. To the top left corner of the app is the navigation bar where the user can move from page to page.

Figure 139, **Custom Drawn Dispense Food Page**
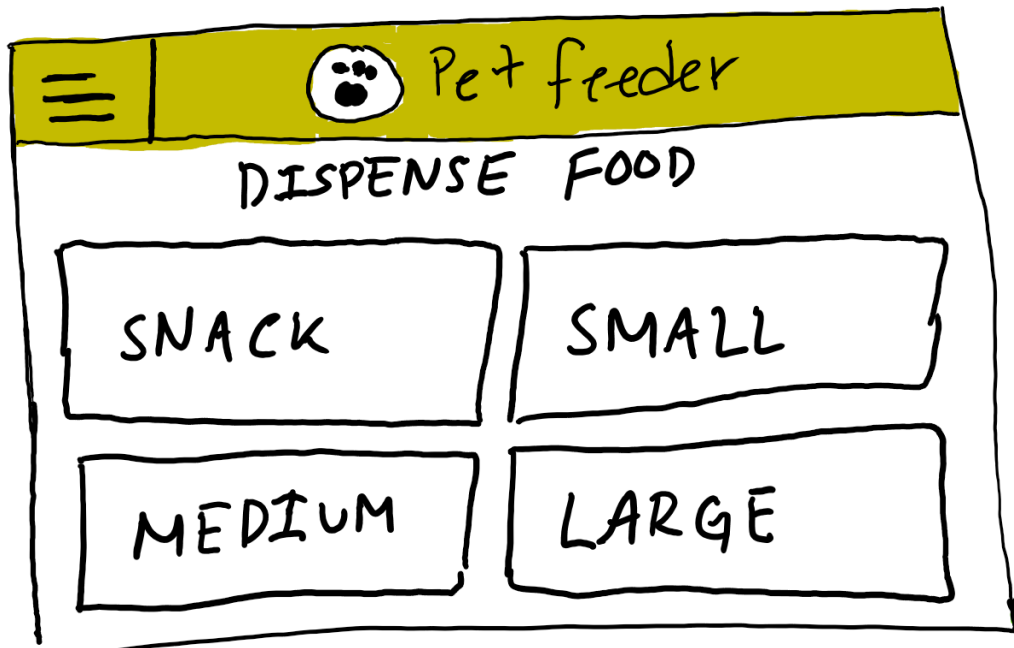
Here is an image of the dispense food page where you can manually dispense the food for your pet and the desired size of the meal.

6. Overall Integration and Testing

This section will cover integrated design with all parts and PCB design.

6.1 Fully Integrated PCB Design
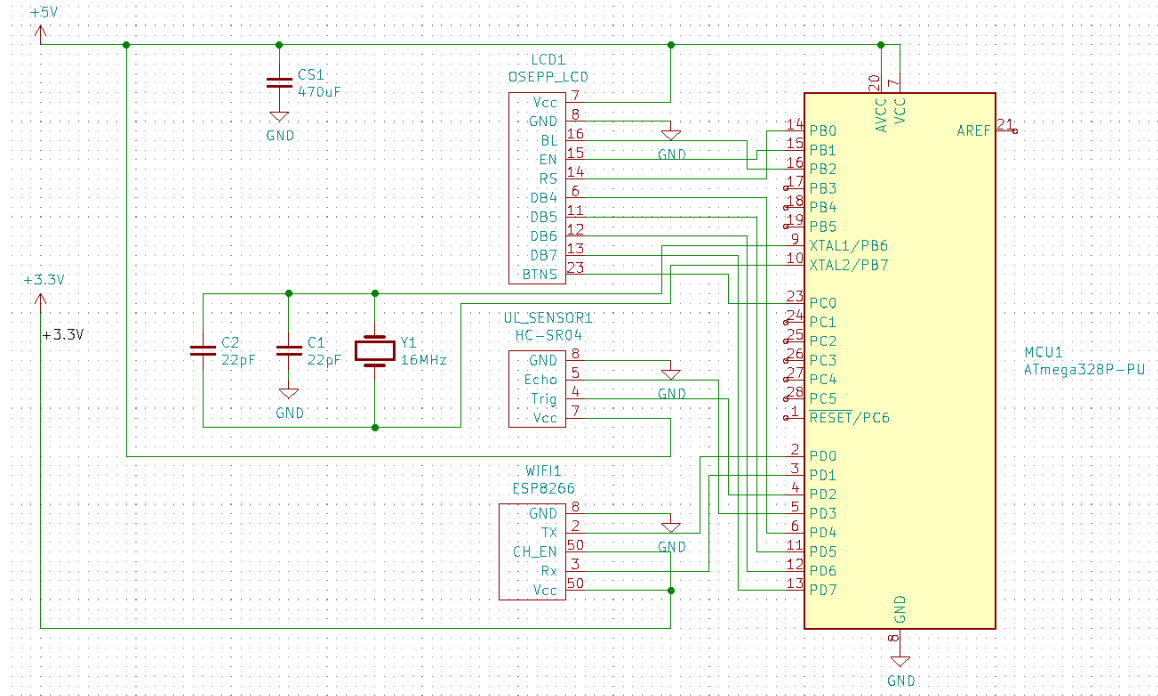
Figure 140, **Fully Integrated PCB Design Schematic**

The schematic above depicts a fully integrated PCB design which incorporates all the components of the entire design.

The power section is shown in the upper left of the schematic. It takes an input from a plug-in power source and regulates the voltages to 5V and 3.3V. From there, it is used by the corresponding components that use the specified voltages from their respective datasheets.

The LCD is represented in the schematic and connects to the pins as needed. This module has priority of pin selection over the other components as it uses the most pins in the design. From there, the remaining pins were selected for the remaining components.

The remaining components, the ultrasonic sensor, wifi module, and weight sensor, use the remaining pins as they use a range of 3-5 pins depending on the module.

The weight sensor is shown connected to the load cell in the schematic for representation purposes but was not included in the PCB design as it is an external component which connects to the weight sensor module directly and does not connect to any pins on the board itself.
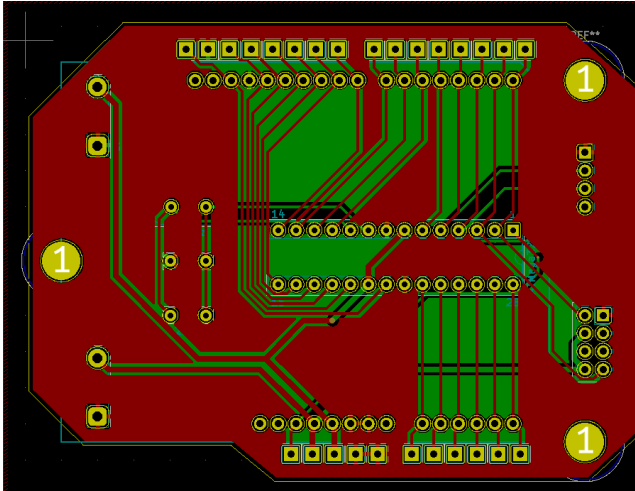
The image on the left depicts the PCB design with the two layers of routes, vias and through hole connections.

The red layer shown is the front layer of the PCB. The lines are routes to various components and the filled, red regions are the ground planes connecting all grounds.

The green layer behind the red layer follows the principles above. The only difference is that the green layer is the rear side of the board where connections and routes go to avoid overlapping of routes as there would be voltage and current shorts and discrepancies.

The images shown below depict a 3D representation of the PCB when fully assembled. This feature is only available in the used EDA KiCad. It can be noted that some pin sockets on the design match the pin sockets on the Arduino Uno board. This was done so that the OSEPP LCD Module can be mounted on the board and its pins used in conjunction.

The LCD shield will be mounted above all the components shown below and will look like the image shown in the LCD Hardware Implementation section of User Control Design. The voltage regulators shown were moved outside of the board to not interfere with the board of the LCD module shield. Also, the pin sockets for the various components were added to allow flexibility of wire in the building phase of the design.

*Figure 142 and 143,* **PCB 3D View** (left and right)
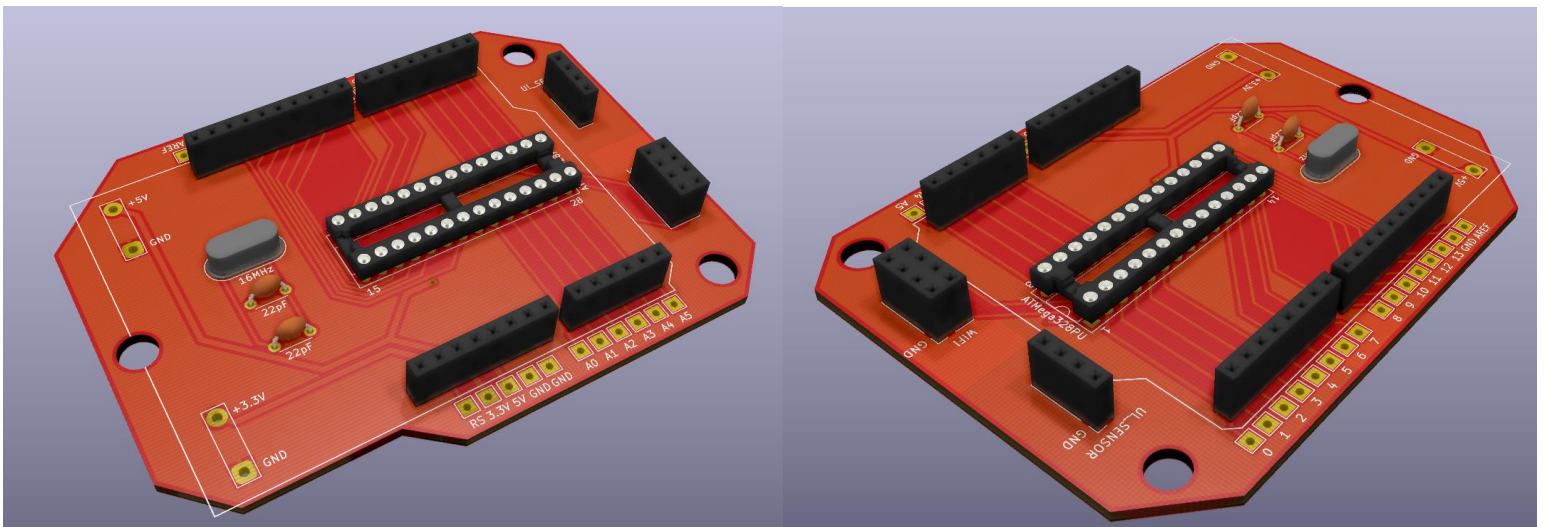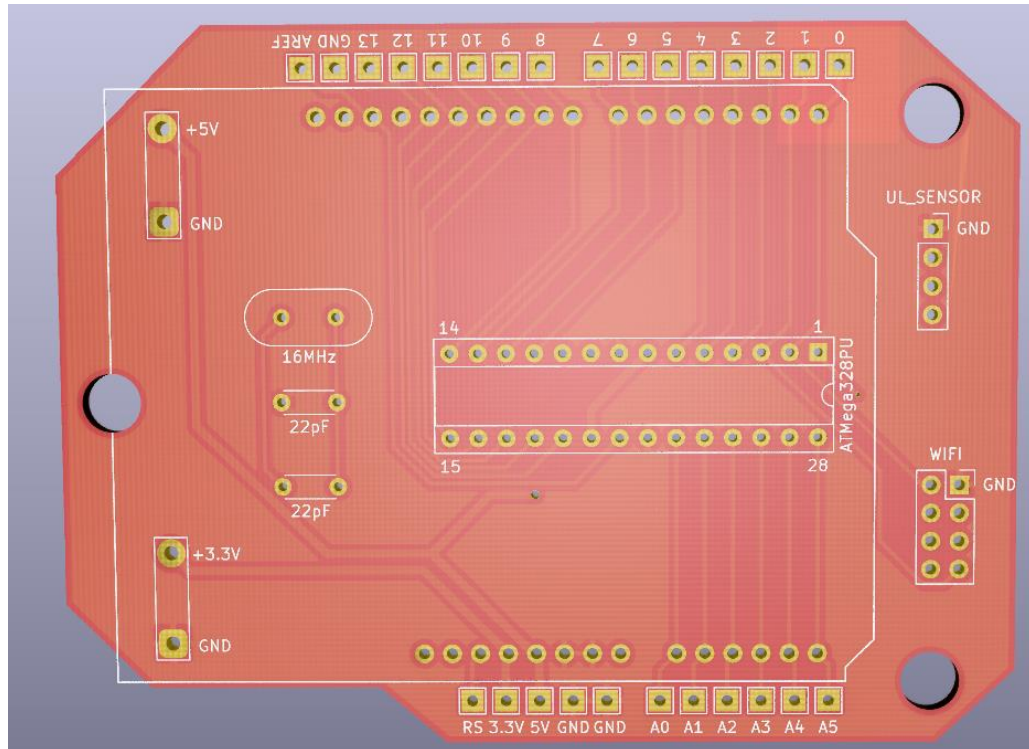
Figure 144, **PCB Without Components**

This is the final representation of the PCB in its raw form with no assembled parts added. Again, it is a 3D representation of the PCB and how it will arrive from the PCB manufacturer provided by an interesting feature within KiCad.

The labeling of components are shown for easy assembly. The labels are subject to change before the final design is completed. They can also be supplemented with more detail to add aid in the assembly process.

Also, for the pin sockets for U2-U4, more detail can be added to further add clarification in the assembly process. The pins will be labeled in more detail before the final assembly and production of the design.

The routes, through holes and vias can also be easily and clearly represented in this image. This KiCad feature has great benefit when it comes to prototyping PCB designs and helping aid in the assembly process.

The routes can be shown with vias that route the route to the other side or layer of the board. Both sides of the board have filled ground planes that are easily discerned from the picture shown above. Through hole design is most beneficial for this prototype design as we have the parts already and have the necessary tools to mount through hole. We don't have the tools needed to mount surface mount parts so through hole design was the next best option for this design. Also, with a through hole design, parts can easily be removed without damaging neighboring components.

7. Administration

This section covers an overview of the project such as budget, problems, and more.

7.1 Milestone Review

This section will break down the milestone completion of the project. It will show the start from the initial idea stage during Senior Design 1 in the Spring 2021 semester down to the final project presentation at the end of Senior Design 2 in the Summer 2021 semester.

Table 5, **Milestone Table (Actual)**

Number	Milestones	Timeframe
Senior Design I		
1	Think of Design Idea/Problem	(1/18/21 – 1/24/21)
2	Finish Divide and Conquer 1.0	(1/25/21 – 1/31/21)
3	**Divide and Conquer 1.0 Meeting (w/ Dr. Lei Wei)**	(2/2/21)
4	Finish Divide and Conquer 2.0	(2/8/21 – 2/14/21)
5	LCD Design	(2/15/21 – 2/21/21)
6	Power and Microcontroller Design	(2/22/21 – 2/28/21)
7	Weight and UltraSonic Sensor Design	(3/1/21 – 3/7/21)
8	Wi-Fi Connection Design	(3/8/21 – 3/14/21)
9	Mobile Application Design	(3/15/21 – 3/21/21)
10	60 page Draft Senior Design I Documentation	(3/29/21 – 4/4/21)
11	**Document Review Meeting (w/ Dr. Lei Wei)**	(4/5/21)
12	Ordering and Acquiring Materials	(4/5/21 – 4/11/21)
13	Testing Individual Parts	(4/5/21 – 4/11/21)
14	100 page report submission_updated	(4/12/21 – 4/18/21)
15	Final Document Due	(4/26/21 – 5/2/21)
Senior Design II		
16	Assemble Prototype	(5/17/21)
17	Testing and Redesign	TBD
18	Finalize Prototype	TBD
19	Peer Report	TBD
20	Final Documentation	TBD
21	Final Presentation	TBD

Our group met all the milestones for Senior Design I. The 60 page draft met its requirements of 60 pages however, there were some problems with formatting and lack of information. In our 100 page report we managed to fix our previous mistakes and reach our 100 page goal. There were a few white space issues, but no other errors or missing information with our submission.

7.2 Budget and Finance

Table 6, **Estimated Budget**

Component	Price (Estimated)
Power Supply / Regulation	$15-$20
MCU	$10
Sensors (scale, level, etc)	$10
WiFi (on device)	$10-$35
LCD	$10-$50
Buttons / Control	<$1
PCB	$5-$10
Mobile App (Android, Apple, Web Interface)	Free
Motors	$15
Housing (plastics, rubber feet)	$10
Pet Food	$50

The prices for the components are based on the prices on seller sites such as Amazon and Digikey. For MCU, a simple microcontroller is Arduino Uno. It goes as high as $20 on Amazon and as low as $10 on Walmart.

A pressure sensor will need to be purchased for the food weight. The pressure sensor's job is to weigh the contents of the food and send a signal back to the MCU to stop dispensing. When using an Arduino Uno MCU the weight signal is low, so it will need to be amplified. The Arduino Uno can convert the signal from digital to analog, but it's very weak. There, Hx711 module sensors are used as load cell conversion to assist with measurements. They are around $6 but usually are provided with load scales.

There are 2 different kinds of load cell scales we can use for this project. One scale is a rectangular shape known as a bar scale: with a long width but shorter height. The other scale is square shape known as a bathroom scale. Degraw sells both of them with an Hx711 amplifier included as well. It is $13 for the bathroom scale and the rectangular load cell. However, the bathroom scale is a set of 4 but the load cell is just 1.

The pet feeder will also need a way to send messages to the user. Wifi will be used to connect the Arduino to the internet so it can connect an application and send notifications.

The ESP8266 wifi module allows the Arduino to be used with Wifi, and it is priced high $20, low $5 sold mostly at $10. A few other parts we looked into allowing the arduino uno to connect to wifi were the MKR1000, MKR1010, ESP32, and the Arduino Uno with integrated wifi. In respective order they cost on average at $35, $33, $10, and $45. With the prices range being so large it makes us put into perspective the price and the ability each part has to offer to the overall product.

The LCD price can vary depending on the features wanted such as size or color. However, for this project a simple LCD that is compatible with Arduino Uno will suffice. A 16x2 LCD costs around $10. A few other LCDs that are up for consideration for the project are the OSEPP LED, Adafruit LCD, Nokia 5110/3310, SSD 1306 OLED, and Adafruit TFT Touch Shield. The OSEPP LED cost on average about $15, the Adafruit LCD cost on average is also $15, the Nokia LCD is about $10 each, the SSD 1306 OLED comes at $18, and finally the adafruit TFT costs about $40 each. The display is an important part of the design of the product since the user may be using it a lot. The cost of most of the LCD costs around the same price range so selecting a LCD should be easy.

A button will be added for manual food dispensing. This button can be very simple, but of course it needs to work with the Arduino board. On Digikey and TE connectivity, push buttons are very cheap, they can go for $0.10. Many buttons and Keys can be found in different kits with other parts as well, so the cost for these will be very minimal if we needed to order extra buttons separately. The same goes for the LEDs we will be using.

A motor is needed to allow the food to drop. Servo motors are the most common motor to work with Arduino boards and will suffice for this project. They are mid priced at $10, but can go up to $20. The prices for motors are varying still because we still do not know the exact strength of the motor we would like to use for the gate of our project.

Other costs that play into our project that are hard to estimate are the physical materials to build the frame of the device and the tank for the food. Since the technology is not yet selected we would like to save the type of material that we would use to create the frame, for the tank to hold the food however we plan on using some sort of plastic that can be melted and formed or just put together which should not cost much all together. We predict all the parts for the frame and tank should cost about $10.

One of the most expensive parts of the project that is essential for testing is the pet food itself. For our presentation we have selected to buy dog food, the cost of that will come in around $50. There are cheaper alternatives but we wanted to get the high price on the food to see the max cost of the product to see how much will need to be spent to create and test the product.

The budget of our project is on the lower end because everything we order will have to be out of pocket. Being college students is quite straining financially on its own, but now having to do a project that is not sponsored may affect the parts we select because of the financial strain it may cause our group. Most of the estimates are about where they should be but some are more on the high side so we know we have saved enough for the project.

<div align="center">

Table 7, **Actual Budget**

</div>

Component	Price
Power Supply / Regulation	$15
MCU	$22
Sensors (scale, level, etc)	$11
WiFi (on device)	$4
LCD	$15
Buttons / Control	$1
PCB	$5
Mobile App (Android, Apple, Web Interface)	Free
Motors	$5
Housing (plastics, rubber feet)	$50
Pet Food	$25
Total	**$153**

We can see from the table above that a few of the prices we have listed are higher than the predicted cost of the parts, and there are some prices that are much lower that the parts prices that were predicted before. We see that the grand total spend on the project is $153.

Some of the reasons the prices have raised compared to the original estimate is because of the vendor from which they were bought from. Initially we were trying to find the lowest price possible, and possibly forgetting to find a dependable source to buy it from. Also the price may have changed depending one who could get the products here faster, for the breadboard testing we needed to get the parts in as fast as possible.

A few reasons some of the prices are much lower than the predicted ones are because in this table we show the price for one component of each part. Some parts are sold in bundles and are charged more. For example the wifi modules cost $16 for 4 modules but, we only needed one so only $4 is shown on the table.

The prices we see here in the actual budget can be minimized if we planned on manufacturing the actual product, because a lot of these parts can be mass produced at a cheaper price, but since we are only making one device it cost a little bit more.

7.3 Project Design Problems

Hardware problems

This section will be focused on the potential issues we encountered throughout the project design. One of the main issues we run into during the design of our project is acquiring the parts at a lower cost, most of the suppliers want to sell large quantities of each item and they also have prices attached to each different quantity. However, we only needed a few items for our project, therefore we had to buy a large quantity that we didn't necessarily need. Another issue during the design phase of the project was finding the comparable parts to our PCB, the packaging of each part was very important in order to get the right parts for the printed circuit board. The research involved when it comes to parts packaging was tedious and time-consuming, however it is very crucial to the success of the project. The packaging of each component for the PCP is either through-hole or SMT (surface mount technology), therefore it is crucial to get this piece of information in order to get the right packaging that would be placed correctly on the PCB.

Another major issue during the design is the research for the available parts since there are very long delays in the manufacturing and delivery of certain parts. The majority of these delays are due to the pandemic since this year and last year most factories went through shut-downs, in fact some parts of the world are still dealing with Lockdowns. Most of the components used in electronic devices are manufactured in different countries therefore the logistics behind manufacturing and delivering these parts during the pandemic was very challenging especially to the end consumer. This was especially challenging for us since we have to decide which part will be used in our project, we have to do a lot of research and investigation concerning the availability and delivery times of the parts. Each time we compile BOM and try to check for pricing availability and delivery time, we always find a couple of parts that are either too expensive or have very long delays that are as long as twenty weeks in some cases. This, of course is not conceivable given that we will start senior design II soon and cannot wait very long since we still have to test and prepare for issues that might arise. Therefore, we have to go back each time and research for the parts that would be delivered in a timely manner with a decent price.

In addition to these hardware issues, we encountered more issues when it comes to manufacturing the PCB schematic design that we built. When investigating different manufacturers, we found out that they have different rules and ways of designing prototype PCBs for their customers. In our case when dealing with JLC for the power supply PCB, we found out that they require a Gerber file to be sent to them in order to build the prototype PCB. Their website was very helpful in providing instructions on how to obtain the Gerber file from EAGLE software. This file along with the BOM that was obtained from the board layout and schematic design were sent to JLC. On their end JLC also performs a first check when they first receive the Gerber file to make sure that the PCB can be built and has no defects. The software or the means they use to check the customers schematics or PCB layout sometimes do not recognize some of the parts that the customer is using, which causes the order to be sent back to the customer to make the necessary corrections that use different parts that they sometimes recommend. This manufacturer's website was very

helpful when it comes to part specifications, they also provide good communication and recommendations if there are any errors.

Wifi Problems

When first selecting our project, one thing we really wanted to have for the automatic pet feeder is a way to control the device remotely, from very far places. The options that we had for connecting remotely were wifi, bluetooth, and zigbee. Bluetooth and zigbee both had issues with range, so we selected wifi. When selecting a MCU that everyone was mostly comfortable with we ended up choosing the arduino uno as our MCU. The issue we had was finding wifi modules that are compatible with the arduino uno, and also not being mounted on a MCU already.

Once we were able to find a few wifi modules that would work for the arduino uno, we came across the issue that not a lot of people ever use the arduino uno for Internet of things projects, so there was not a lot of documentation on getting the wifi module to work with the arduino uno. To fix this problem we had to look at multiple examples of how other wifi modules worked and to the best of our ability try to piece together all the sources of information to get it to work. We seemed to get the wifi to connect to the arduino but sending messages back and forth may still be an issue.

Now having the wifi module situation handled, there was very little to no documentation or examples on how to create a mobile application to be able to send information and commands to the device. With many of our members having little to no experience when it comes to mobile application development, and very little experience with sending data through the internet this was a very big challenge for us. After looking at both aspects individually and finding documentation on how to create a mobile application, and sending data through the internet. Since we have not fully developed the mobile application yet we are hopeful the code we plan on using will work but it still may be a major obstacle in our product development.

Integration Problems

When integrating all our parts together it was a little bit of a mess since we had so many components coming together and connecting to the arduino uno. There were times when a few parts used the same digital pin, which would cause a problem because of information clashing and causing the wrong things to happen. A way we went about to fix this was finding all the digital pins on the board and finding out which ones can be used for external components and re-wired some of the components to different digital pins, so each part can have their own pins so no data being sent to the pins would clash with another part. The only pins we allowed to share oare the power pins because there is a common ground for all parts and a common Vcc for the 3.3V and 5V power pins. This may change depending on the power supply for each of the parts.

7.4 Project Roles

For the majority of this project we stayed faithful to our project's block diagram when deciding the roles. Nick is in charge of LCD and PCB design, Hamid is in charge of power supply, Mehrob is in charge of micro controller LED, and sensors, and Bao is in charge of Wi-Fi and mobile integration. Each member was responsible for researching his own part. After research prices, specifics, abilities, etc we would discuss the part in a meeting.

Each member had to research their part and also report it in the documentation. Any pricing, datasheets, problems, or diagrams was the responsibility of the group member. As stated in the milestone review, every group member was successfully able to research and write about their part.

Nick's role of LCD including his comparison of all 4 parts. His research included pricing, compatibility with Arduino, and specifications such as size. Nick was also in charge of PCB design and production. He designed the schematic to incorporate all of our parts, so his role was very significant.

Hamid's role was with power; all components had to make sure they got enough power from an outside source. His role is very critical to make sure all of our components could be powered together and run continuously.

Mehrob's role was microcontroller and the accessories that go with it such as LED, sensors, and motors. He had to research a MCU that could achieve all the goals and requirements for this project. The MCU had to also be within budget and be usable with all the components.

Bao's role was Wi-Fi integration and mobile application. He had to research the different kinds of Wi-Fi shields for the Arduino so it could be a transceiver and receiver. The mobile application requires functionality and interface design.

7.5 Planning Ahead

Planning ahead for the group, we need to get other pieces for our pet feeder such as casing and the container built. The PCB is fully designed but just needs to be ordered. We have the final design but it needs to be constructed. All of the other components have been tested to work individually but need to go together.

Along with the parts, the code for the MCU needs to be put together as they currently work individually. As a group we need to plan out the code and write it all out. Our code is currently individual, so we need to merge it together.

We need to meet as a group to build the prototype. The first prototype should be able to do most of the functions such as detection of the food and open/close the dispenser. We also need to routinely meet to test the prototype for error and discrepancies.

8. Conclusion

We have covered a lot of information in this paper, it is all about how we plan on building our automatic pet feeder. The reason we chose this project was because many individuals have pets and need a way to feed them when they are away so we are here to address this problem. The way we plan to fix this problem is an automatic pet feeder that a pet owner can control from wherever they are. To do this we plan on making an automatic pet feeder that users can set timers to allow food to dispense or manually dispense food from their mobile device from a companion application. Our device will also have the ability to portion meal sizes to meet the needs of their pet.

We first started off by researching each of our parts and comparing different technologies and different parts to see which one would best fit our product. After comparing and careful consideration we then selected our parts, and went into detail about how we plan on using it in our design and the importance of it. We then ordered the parts for the breadboard testing since things may take some time to arrive due to COVID-19 and high mailing volume.

Following are the Standards and constraints, here we discussed the what standards we are planning to abide by when creating the product and what might be some major constraints to our product. Some standards we covered were coding development and planning standards and power standards. Some constraints we have are time, finance, and the ability to get certain materials.

After discussing how we plan on designing we then go into the design phase which is where each of us designed how each of the parts should look and work individually. Once designed we then tested our design using breadboard testing to confirm our hardware and software was working. After we finished testing each component separately we then moved on to an integrated design where all of our components came together to the microcontroller. Once finished with the design we then proceeded to the PCB design to mount all of the parts.

Finally we then covered our milestones and seeing if we made all of them, we seem to do pretty well on making sure everything was turned in, but we did have a few mistakes on what was supposed to be covered in each section, but after the first review we then fixed all of our mistakes by the next check in. We also discussed our estimated cost versus our actual cost after ordering most of the parts. This section covered any problems we had while researching and designing the product as well. Project roles and what each member was in charge of is also covered. Lastly we have the planning ahead section where we reflect on how we have done so far and what needs to be done in Senior Design 2 to be more successful than this semester.

Our goal of this product is to create something that would solve an issue that we have in society today. The issue is finding a way to allow pet owners to feed their pets even when they are far from home. It is to make the lives of all pet owners easier and make them feel more secure about living their lives.

Appendix
A1. Reference

SparkFun Electronics. (n.d.). *SparkFun Breadboard Power Supply 5V/3.3V*. PRT-00114 - SparkFun Electronics. https://www.sparkfun.com/products/114.

Gupta, S. (2020, June 3). *Design your own Compact 5V/3.3V SMPS Circuit for Embedded and IoT Projects*. Circuit Digest. https://circuitdigest.com/electronic-circuits/design-your-own-compact-5v-33v-smps-circuit-for-embedded-iot-projects.

User, S. (n.d.). Home. Retrieved April 27, 2021, from https://www.osepp.com/electronic-modules/shields/45-16-2-lcd-display-keypad-shield

Schematic: https://www.osepp.com/downloads/pdf/OSEPP_LCD_Keypad_Shield_V1-0-SCH1.pdf

Datasheet:
https://www.osepp.com/downloads/pdf/SPLC780D_DS.pdf

Drtorq. (2019, May 03). Off-The-Shelf hacker: The art of soldering. Retrieved April 27, 2021, from https://thenewstack.io/off-the-shelf-hacker-the-art-of-soldering/

Main Page:Web_page_person. (2021, January 21). Login form example in Kotlin Android. Retrieved April 27, 2021, from https://www.tutorialkart.com/kotlin-android/login-form-example-in-kotlin-android/

Industries, A. (n.d.). LCD shield Kit w/ 16x2 character display - only 2 pins used! Retrieved April 27, 2021, from https://www.adafruit.com/product/772#technical-details

Ada, L. (n.d.). Nokia 5110/3310 monochrome lcd. Retrieved April 27, 2021, from https://learn.adafruit.com/nokia-5110-3310-monochrome-lcd

Tripoli, S., Santos, R., John, Marin, C., Bello, S., Gorki, . . . Sriram. (2020, July 30). Guide for i2c oled display with arduino. Retrieved April 27, 2021, from https://randomnerdtutorials.com/guide-for-oled-display-with-arduino/

Industries, A. (n.d.). 2.8" TFT TOUCH shield for Arduino with resistive touch screen. Retrieved April 27, 2021, from https://www.adafruit.com/product/1651#technical-details

LiquidCrystal. (n.d.). Retrieved April 27, 2021, from https://www.arduino.cc/en/Reference/LiquidCrystal

Santos, S., Says, J., Jimmy, Says, J., James, Says, M., . . . Says, S. (2020, May 18). Esp32 vs esp8266 - pros and cons. Retrieved April 27, 2021, from https://makeradvisor.com/esp32-vs-esp8266/

Imjeffparedes, & Instructables. (2018, March 27). Add wifi to arduino uno. Retrieved July 25, 2021, from https://www.instructables.com/Add-WiFi-to-Arduino-UNO/#discuss

U. (n.d.). Arduino MKR1000 WIFI. Retrieved April 27, 2021, from
https://store.arduino.cc/usa/arduino-mkr1000

U. (n.d.). Arduino MKR WiFi 1010. Retrieved April 27, 2021, from
https://store.arduino.cc/usa/mkr-wifi-1010

U. (n.d.). ESP8266 pinout, PIN Configuration, Features, example CIRCUIT & Datasheet.
Retrieved April 27, 2021, from https://components101.com/wireless/esp8266-pinout-
configuration-features-datasheet

Paredes, J. (2017, July 6). Add wifi to arduino uno. Retrieved April 27, 2021, from
https://create.arduino.cc/projecthub/imjeffparedes/add-wifi-to-arduino-uno-663b9e

Grokhotkov, I. (n.d.). Esp8266wifi library¶. Retrieved April 27, 2021, from
https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/readme.html

Android studio. (2021, April 24). Retrieved April 27, 2021, from
https://en.wikipedia.org/wiki/Android_Studio#/media/File:Android_Studio_4.1_screensh
ot.png

Android studio. (2021, April 24). Retrieved April 27, 2021, from
https://en.wikipedia.org/wiki/Android_Studio

Web_page_person. (2021, January 21). Login form example in Kotlin Android. Retrieved
April 27, 2021, from https://www.tutorialkart.com/kotlin-android/login-form-example-in-
kotlin-android/

Editor, A. (2017, May 03). Advantages and disadvantages of zigbee. Retrieved April 27,
2021, from https://www.polytechnichub.com/advantages-disadvantages-zigbee/

Wireless technology. (n.d.). Retrieved April 27, 2021, from
https://www.nibusinessinfo.co.uk/content/pros-and-cons-wireless-networking

Robot Research Lab. (2020). *How to use HX711 with Four Load Cells*. *YouTube*.
YouTube. https://www.youtube.com/watch?v=dNiVZBTvwxs.

Williams and Wang, LLC. (n.d.). HC-SR04 Ultrasonic Distance Sensor Module.
https://www.bananarobotics.com/shop/HC-SR04-Ultrasonic-Distance-
Sensor#:~:text=The%20easy%20way%20to%20read,centimeters%20or%20about%201.7
%20centimeters.

N/a. (n.d.). *Arduino - Ultrasonic Sensor*. Tutorialspoint.
https://www.tutorialspoint.com/arduino/arduino_ultrasonic_sensor.htm#:~:text=The%20
HC%2DSR04%20ultrasonic%20sensor,or%201%E2%80%9D%20to%2013%20feet.

Kiani, H. H. (2020, September 13). *Interface HX711 Load Cell Module w/ Arduino:
Build a Digital Weight Scale*. Electropeak. https://electropeak.com/learn/interface-hx711-
load-cell-amplifier-module-arduino-digital-weight-scale/.

https://www.raspberrypi.org/documentation/hardware/raspberrypi/power/README.md

https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf

Abichar, Z. (2018). *Lab Manual for Eel 4742C Embedded Systems*. University of Central Florida Computer Science and Engineering.

esp32-wroom-32_datasheet_en.pdf

Raspberry Pi LTD Trading. (n.d.). *Raspberry Pi 4 Model B Product Brief*. Raspberry Pi 4 Computer Model B. https://datasheets.raspberrypi.org/rpi4/raspberry-pi-4-product-brief.pdf.

https://www.microchip.com/wwwproducts/en/ATmega328P

Atmel. (n.d.). *8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash*. ATmega328P. http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf.

Smith, E. (1970, January 1). Arduino Uno vs TI LaunchPad (MSP430 edition). http://humboldtmcu.blogspot.com/2014/06/arduino-uno-vs-ti-launchpad-msp430.html.

Allan, A. (2014, February 7). Which Board is Right for Me? https://makezine.com/2014/02/07/which-board-is-right-for-me/#:~:text=The%20major%20difference%20between%20LaunchPad,USB%20for%20only%20%242%20more.

Teja, R. (2021, April 5). *What are the differences between Raspberry Pi and Arduino?* Electronics Hub. https://www.electronicshub.org/raspberry-pi-vs-arduino/#:~:text=Raspberry%20pi2%20has%202%20packs,32kb%20of%20storage%20on%20board.

IEEE 1008-1987 - IEEE standard for software unit testing. (n.d.). Retrieved April 27, 2021, from https://standards.ieee.org/standard/1008-1987.html

IEEE/ISO/IEC 24748-2-2018 - ISO/IEC/IEEE international standard - systems and software ENGINEERING--REQUIREMENTS for managers of information for users of systems, software, and services. (n.d.). Retrieved April 27, 2021, from https://standards.ieee.org/standard/24748-2-2018.html

IEEE/ISO/IEC p24748-6 - ISO/IEC/IEEE draft standard - systems and software engineering -- life cycle management -- Part 6: Systems and software integration. (n.d.). Retrieved April 27, 2021, from https://standards.ieee.org/project/24748-6.html

IEEE 802E-2020 - IEEE recommended practice for privacy considerations for IEEE 802(r) technologies. (n.d.). Retrieved April 27, 2021, from https://standards.ieee.org/standard/802E-2020.html

[0a-esp8266ex_datasheet_en](#)

IEEE/ISO/IEC 14764-2006 - ISO/IEC/IEEE international standard for software engineering - software life Cycle processes - Maintenance. (n.d.). Retrieved April 27, 2021, from https://standards.ieee.org/content/ieee-standards/en/standard/14764-2006.html