



Automatic Pet Food Dispenser

Group 25

Members:

Nick Nabors, CpE

Bao Nguyen, CpE

Mehrob Farhangmehr, CpE

Hamid Iglou, EE

Senior Design 1 - EEL4719 - SPRING 2021

Due: April 2, 2021

Project Description

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.

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.

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.

Table 0, Specifications

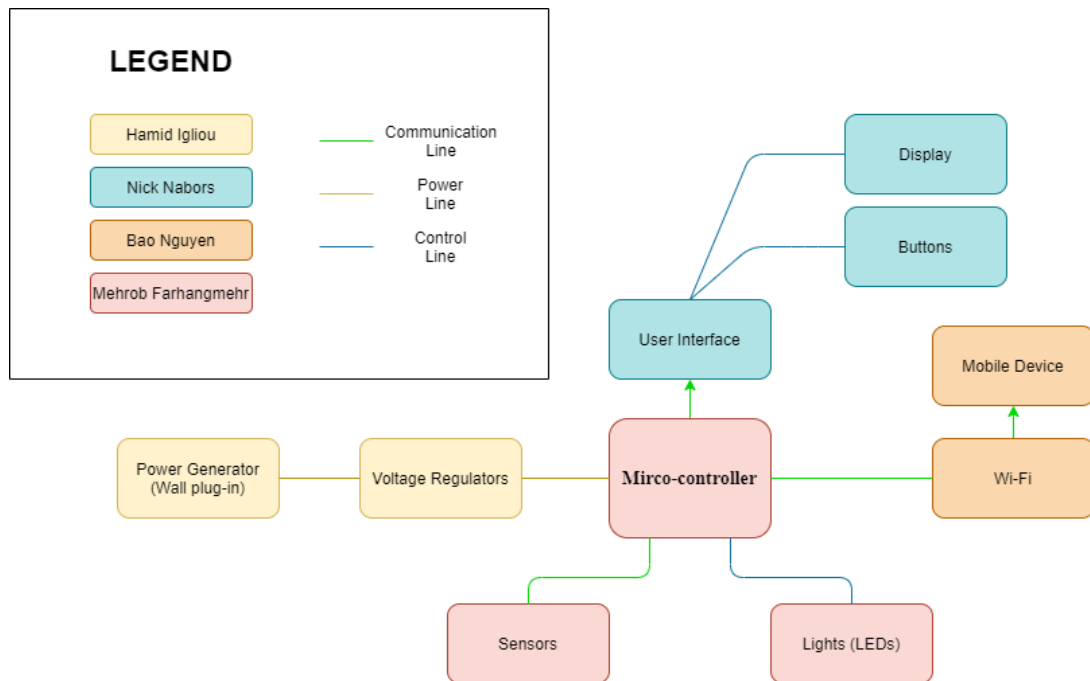
	ADULT DOG SIZE DRY FOOD FEEDING AMOUNT (CUPS)	
Portion Control	3-12 lbs.	1/3 to 1 cup
	13-20 lbs.	1 to 1 1/3 cups
	21-35 lbs.	1 1/3 to 2 cups
	26-50 lbs.	2 to 2 2/3 cups
	51-75 lbs.	2 2/3 to 3 1/3 cups
	76-100 lbs.	3 1/3 to 4 1/4 cups
	Over 100 lbs.	4 1/4 cups plus 1/4 cup for each 10 lbs. of body weight 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 1/4 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, (1/4) Low) ⇒ (3: Red, Empty)	
Weight	2-3 lbs without food	
Container size	Should hold 4 lbs of food	

Here in *Table 0*, 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.

Block Diagram

Figure 0, Block Diagram



Here in *Figure 0*, 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 0*, 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.

Project Budget and Financing

Table 1, Estimated Budget

Component	Price (Estimated)
Power Supply / Regulation	\$15-\$20
MCU	\$10
Sensors (scale, level, etc)	\$10
WiFi (on device)	\$10-\$20
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. When using an Arduino Uno MCU the weight signal is low, so it will need to be amplified. Hx711 module sensors are used as load cell conversion to assist with measurements. They are around \$6. The portable scale itself is \$4.

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.

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 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.

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.

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.

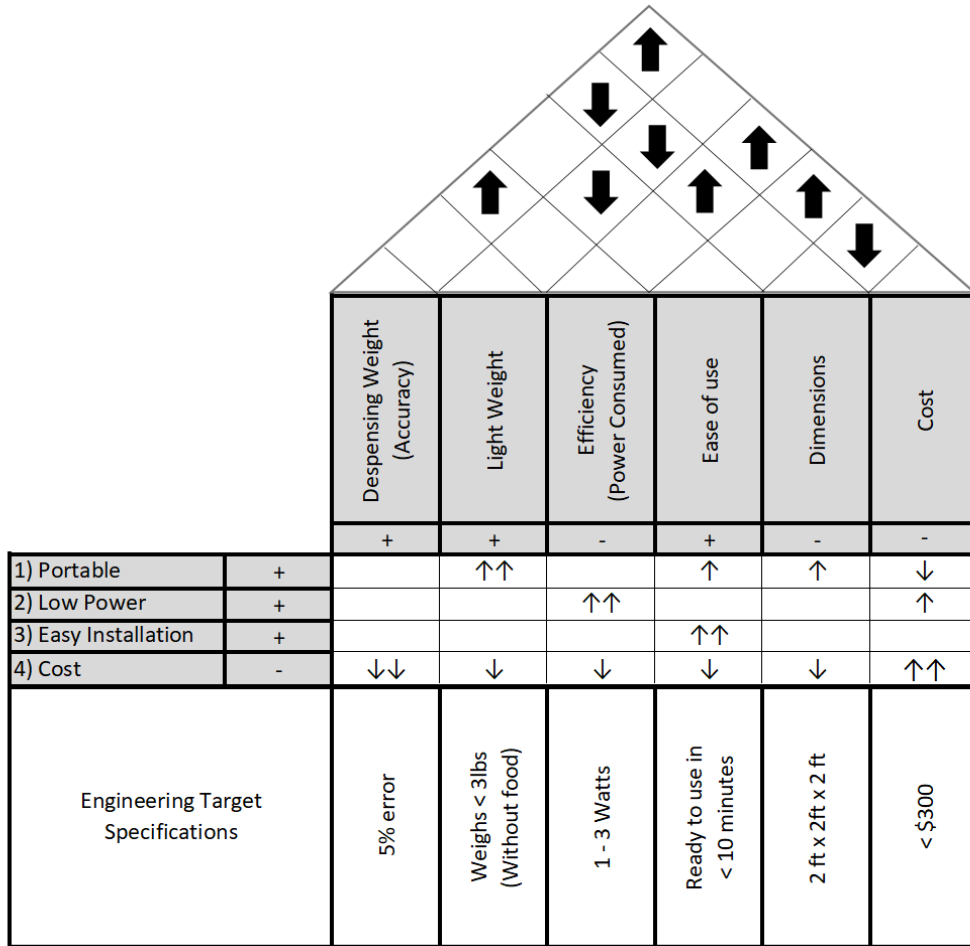
Table 3, Senior Design 2 Milestones (Tentative)

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 2, is that we begin building from our research and budget. The administrative work laid out in *Table 2* will help us reach our end goal of having a successful and completed project.

During the Senior Design 2 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 2. We aim to finish towards the tail end of the semester so we can finalize reports and anything else we need.

Figure 1, House of Quality



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 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.

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.

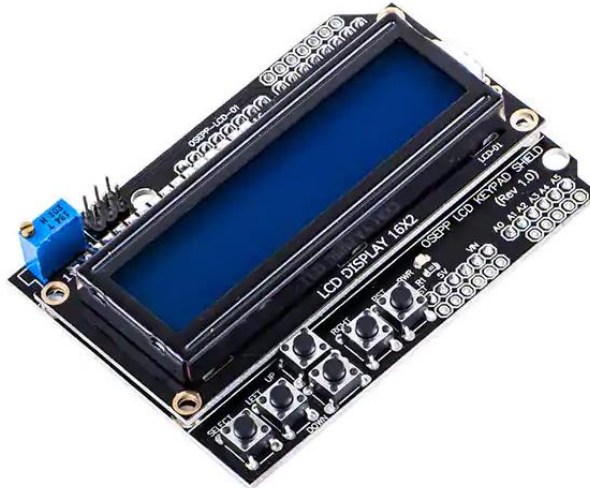
User Interface and Control General Breakdown:

- 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
 - Numpad
 - 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.

Figure 2, OSEPP Electronics LTD, 16X2SHD-01 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 3, LCD Button Schematic

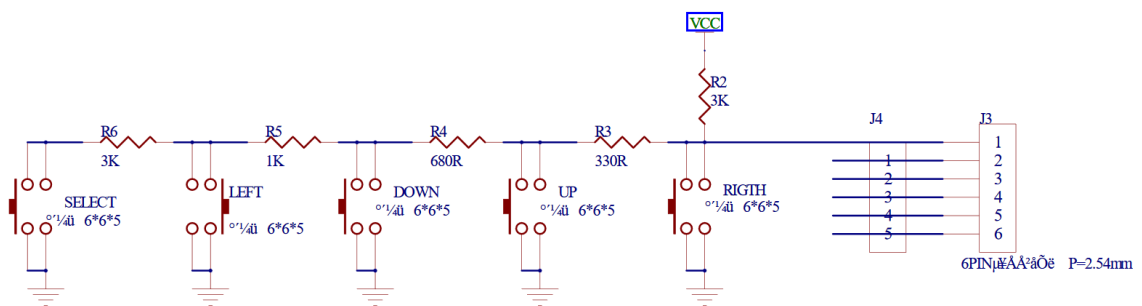


Figure 3 is the schematic for the buttons of the LCD driver/control board of the 16X2SHD-01 LCD Module. It is set up in a series fashion and is directly integrated into the board without any further need to attach any extra components. This helps provide a cheap method of control that will be easy to integrate into our Arduino ecosystem and will help us control the functions of the system as a whole.

Figure 4 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.

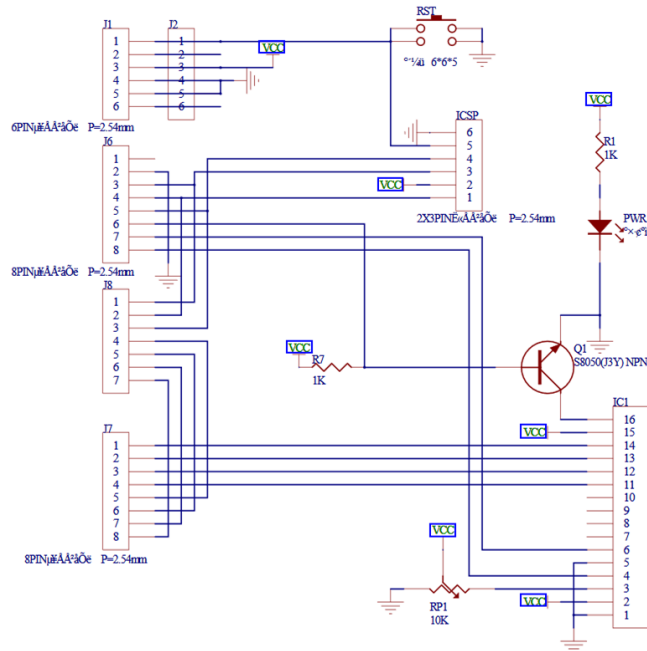
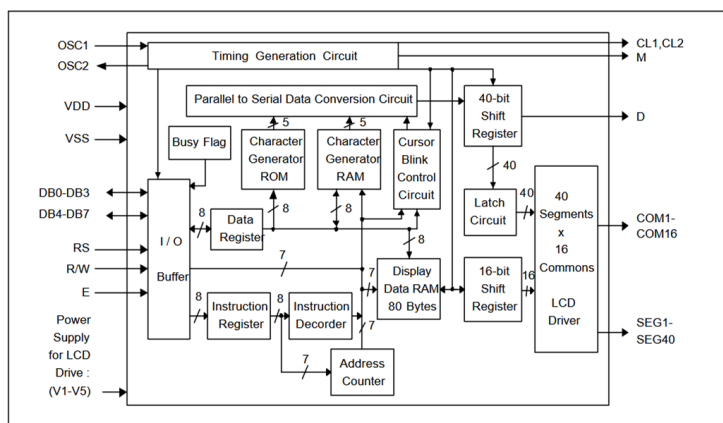


Figure 4, LCD Schematic

Figure 5 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 5, 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 6, 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 external clock operation, the clock is input to OSC1.
OSC2	25		
V1 - V5	26 - 30	I	Supply voltage for LCD driving.
E	38	I	A start signal for reading or writing data.
RW	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 6*. 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 7* 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 that are present. This is obvious from column three, for example. All the numbers are in the third column and so on.

		Higher 4-bit (D4 to D7) of Character Code (Hexadecimal)															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Lower 4-bit (D0 to D3) of Character Code (Hexadecimal)	0	CG RAM (1)															
	1	CG RAM (2)															
	2	CG RAM (3)															
	3	CG RAM (4)															
	4	CG RAM (5)															
	5	CG RAM (6)															
	6	CG RAM (7)															
	7	CG RAM (8)															
	8	CG RAM (1)															
	9	CG RAM (2)															
	A	CG RAM (3)															
	B	CG RAM (4)															
	C	CG RAM (5)															
	D	CG RAM (6)															
	E	CG RAM (7)															
	F	CG RAM (8)															

Figure 7, Character Code Combinations

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.

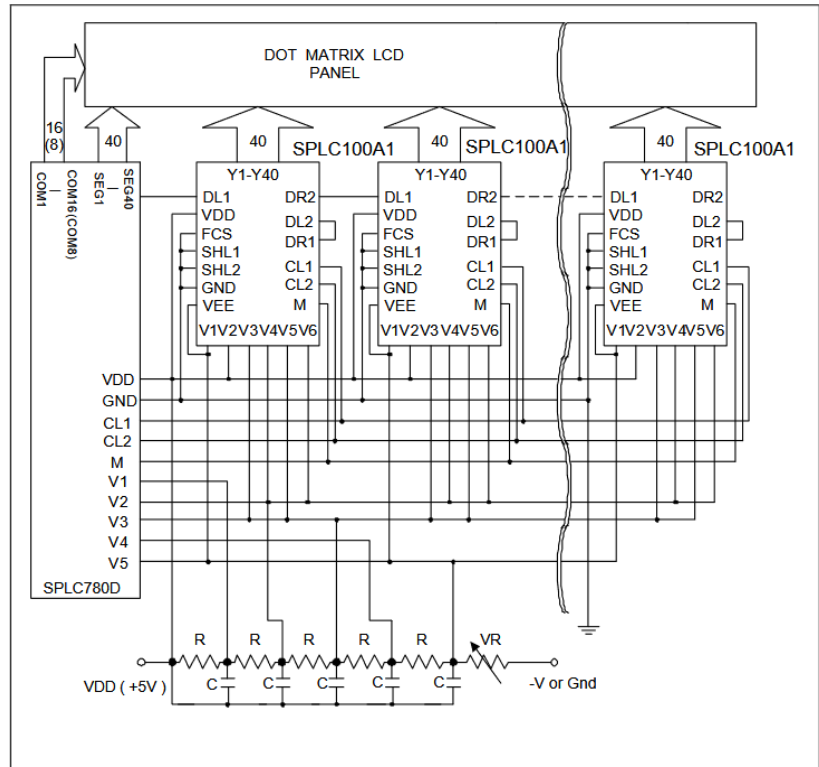


Figure 8, SPLC780D Application Circuit

The schematic presented in *Figure 8* 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.

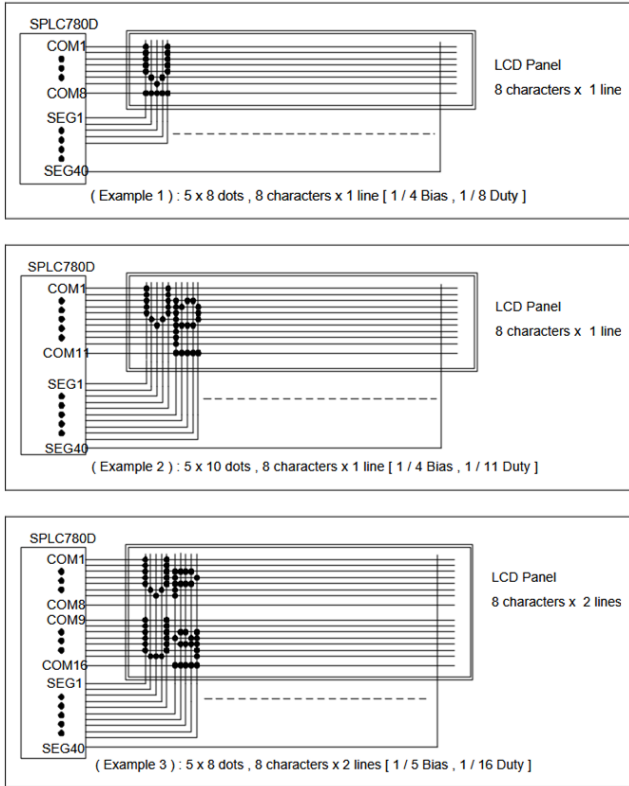


Figure 9, SPLC780D LCD Applications

Figure 9 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 10, 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.

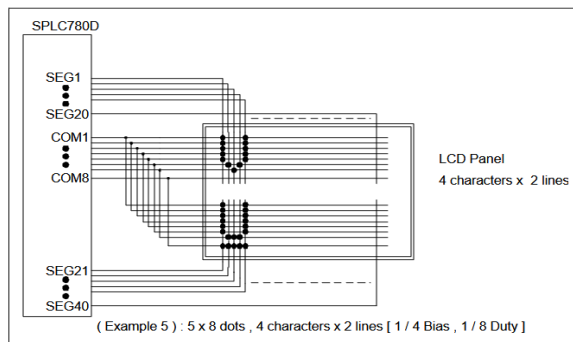
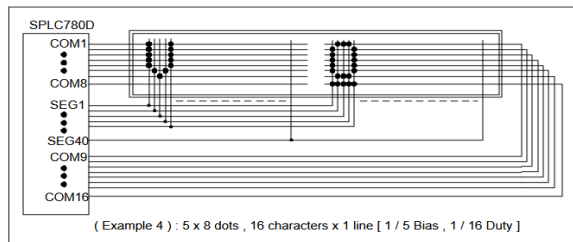


Figure 10, SPLC780D LCD Applications, cont.

In *Figure 12* 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 12*.

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 12, SPLC780D 8-Digit 1-Line

No.	Instruction	Display	Display	Operation
1	Power on. (SPLC780D starts initializing)			Power on reset. No display.
2	Function set RS RW 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 0 1 1 1 0			Display on. Cursor appear.
4	Entry mode set 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.
5	Write data to CG RAM / DD RAM 1 0 0 1 0 1 0 1 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 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 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 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 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 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 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 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 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 0 1 1 1 0		OMPANY_	Write " N ". The display moves to the left.
17	Cursor or display shift 0 0 0 0 0 0 1 1 1 X X		OMPANY_	Shift the display and the cursor's position to the right.
18	Cursor or display shift 0 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 0		OMPANY_	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 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 <table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td></tr></table>	0	0	0	0	1	0		Set to 4-bit operation.						
0	0	0	0	1	0										
3	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>X</td><td>X</td></tr></table>	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.
0	0	0	0	1	0										
0	0	0	0	X	X										
4	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td></tr></table>	0	0	0	0	0	0	0	0	1	1	1	0		Display on. Cursor appears.
0	0	0	0	0	0										
0	0	1	1	1	0										
5	<table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td></tr></table>	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.
0	0	0	0	0	0										
0	0	0	1	1	0										
6	<table border="1"><tr><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td></tr></table>	1	0	0	1	0	1	1	0	0	1	1	1		Write " W ". The cursor is incremented by one and shifted to the right.
1	0	0	1	0	1										
1	0	0	1	1	1										

Figure 13, SPLC780D 8-Digit 1-Line Display

In Figure 13 above, the same process from Figure 12 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 12. This figure, Figure 13, also shows eight digits for a one 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 <table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>X</td><td>X</td></tr></table>	0	0	0	0	1	1	0	X	X		Set to 8-bit operation and select 2-line display line and 5 x 8 dot character font.	
0	0	0	0	1	1	0	X	X					
3	Display on / off control <table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td></tr></table>	0	0	0	0	0	0	1	1	1	0		Display on. Cursor appear.
0	0	0	0	0	0	1	1	1	0				
4	Entry mode set <table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td></tr></table>	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.	
0	0	0	0	0	0	1	1	0					
5	Write data to CG RAM / DD RAM <table border="1"><tr><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td><td>1</td></tr></table>	1	0	0	1	0	1	0	1	1	1		Write " W ". The cursor is incremented by one and shifted to the right.
1	0	0	1	0	1	0	1	1	1				
6	:	:	:										
7	Write data to CG RAM / DD RAM <table border="1"><tr><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td></tr></table>	1	0	0	1	0	0	0	1	0	1		Write " E ". The cursor is incremented by one and shifted to the right.
1	0	0	1	0	0	0	1	0	1				
8	Set DD RAM address <table border="1"><tr><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table>	0	0	1	1	0	0	0	0	0	0		It sets DD RAM's address. The cursor is moved to the beginning position of the 2nd line.
0	0	1	1	0	0	0	0	0	0				
9	Write data to CG RAM / DD RAM <table border="1"><tr><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td></tr></table>	1	0	0	1	0	1	0	1	0	0		Write " T ". The cursor is incremented by one and shifted to the right.
1	0	0	1	0	1	0	1	0	0				
10	:	:	:										
11	Write data to CG RAM / DD RAM <table border="1"><tr><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>0</td></tr></table>	1	0	0	1	0	1	0	1	0	0		Write " T ". The cursor is incremented by one and shifted to the right.
1	0	0	1	0	1	0	1	0	0				
12	Entry mode set <table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td></tr></table>	0	0	0	0	0	0	0	1	1	1		When writing, it sets mode for the display shift.
0	0	0	0	0	0	0	1	1	1				
13	Write data to CG RAM / DD RAM <table border="1"><tr><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>1</td></tr></table>	1	0	0	1	0	1	1	0	0	1		Write " Y ". The cursor is incremented by one and shifted to the right.
1	0	0	1	0	1	1	0	0	1				
14	:	:	:										
15	Return home <table border="1"><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td></tr></table>	0	0	0	0	0	0	0	0	1	0		Both the display and the cursor return to the original position (address 0).
0	0	0	0	0	0	0	0	1	0				

Figure 14, SPLC780D 8-Digit 2-Line Display

Figure 14, is a similar process carried out from Figure 12 and Figure 13 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.

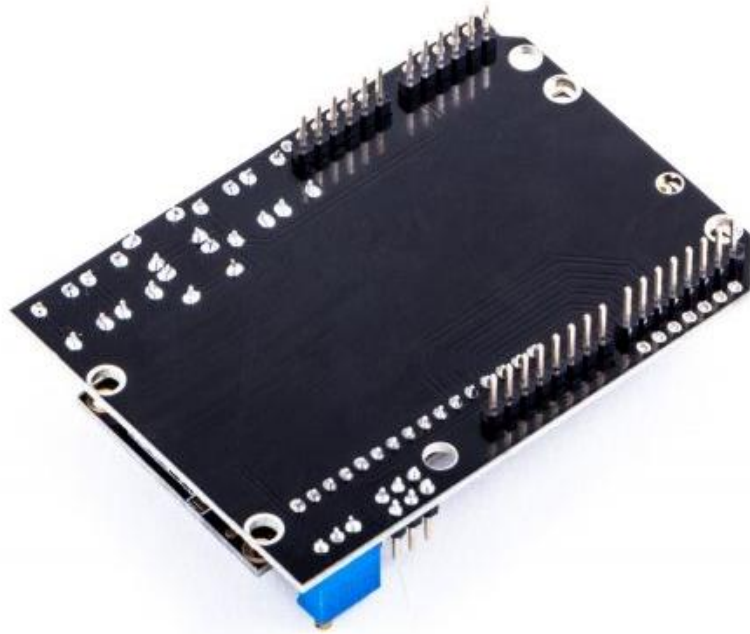
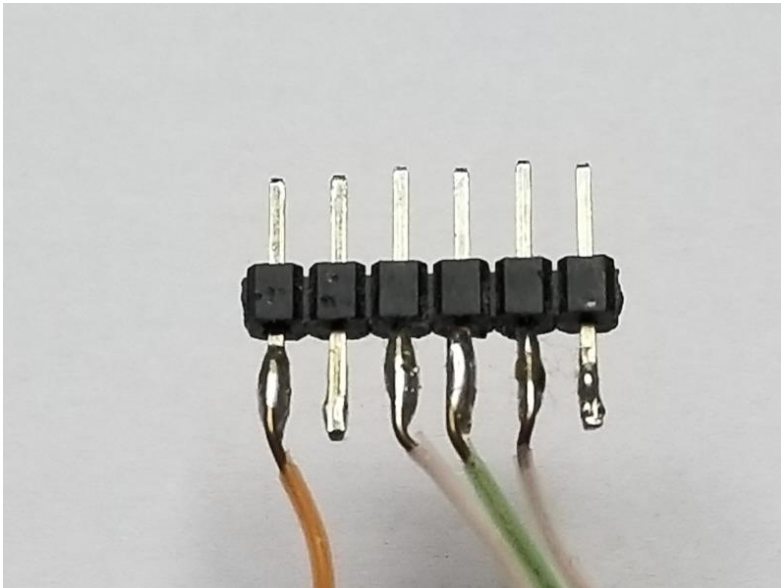


Figure 15, LCD Module Rear Pin View

In *Figure 15*, 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 16*, the process for soldering the rear pins from the LCD module in *Figure 15* 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 16, 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 17, LCD Sample Code

Figure 17 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 Arduino 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 Arduino will determine that the key that was pressed is not a valid key based off of the code compiled to the Arduino.

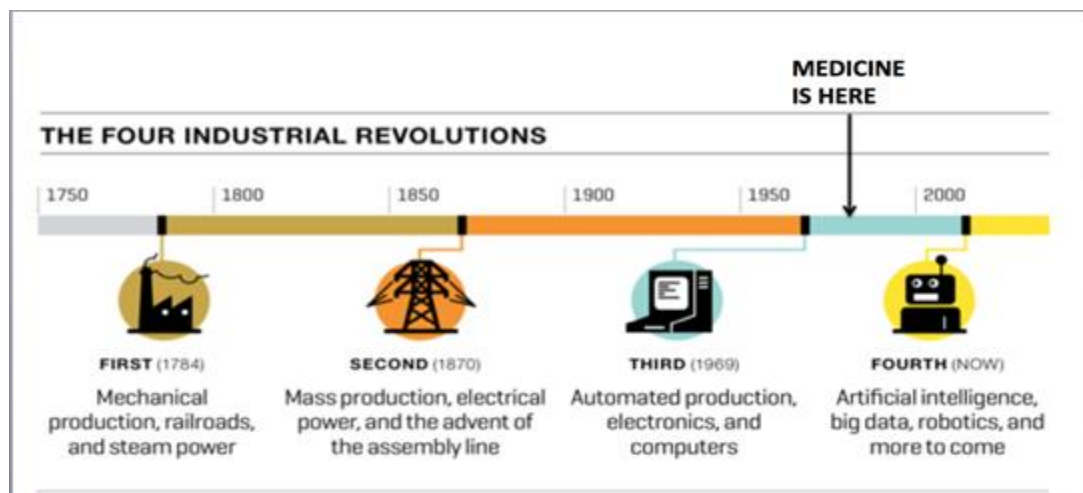
Power supply

1.1: 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 the 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 18, Timeline of industrial revolution



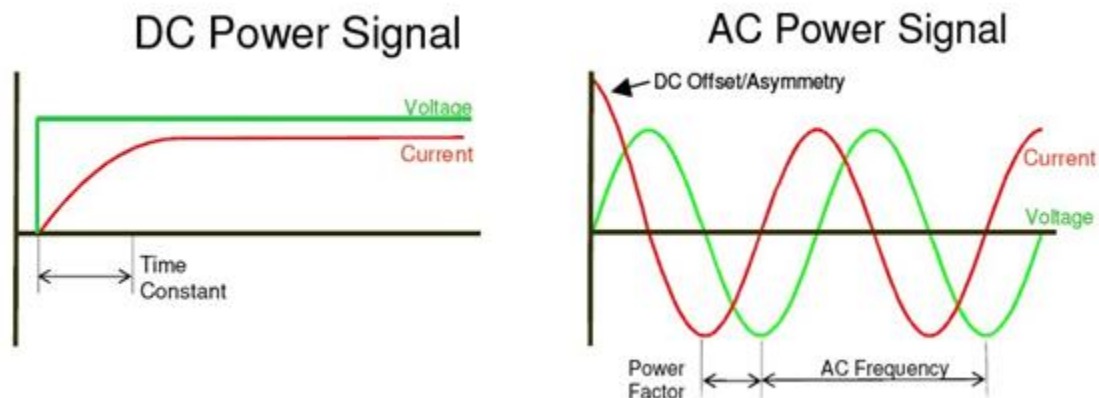
1.2: Power Electronics

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.

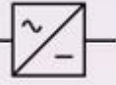

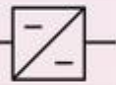
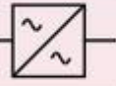
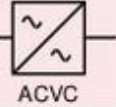
Figure 19, AC vs DC power signal



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 20, Classification of power electronic converters

Converter Type	Input	Output	Symbol
Rectifier	A.C. at constant voltage and frequency	D.C. at variable voltage	
Inverter	D.C. at constant voltage	A.C. at desired voltage and frequency	
Chopper	D.C. at constant voltage	D.C. at desired voltage	
Cycloconverter	A.C. at constant voltage and frequency	A.C. at desired voltage and frequency	
A.C. Voltage Controller	A.C. at constant voltage and frequency	A.C. at desired voltage and input frequency	

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 power supply should also incorporate a filtering capacitor and voltage regulator for a smooth output.

2: 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

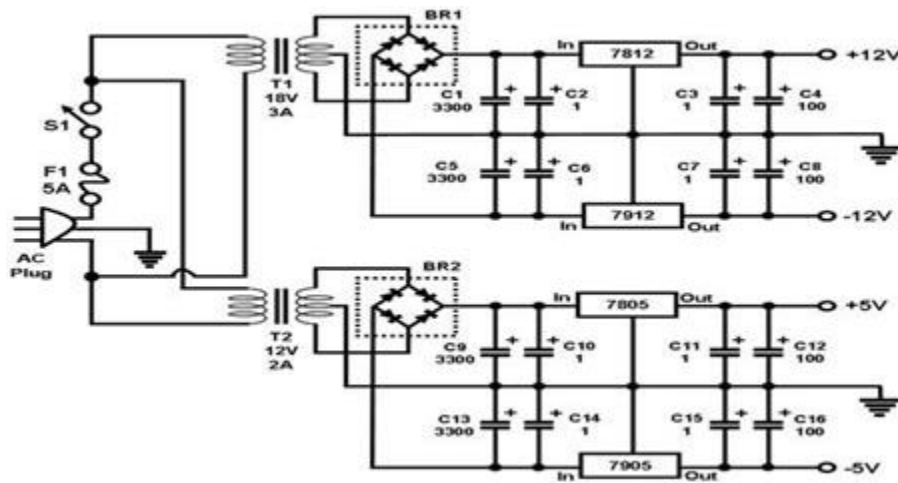
5. Current protection device (fuse)
6. Voltage protection device (varistor)
7. Test equipment

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

2.1: Reference Design

A reference design for a multi-output power supply is presented in the following figure:

Figure 21, 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.

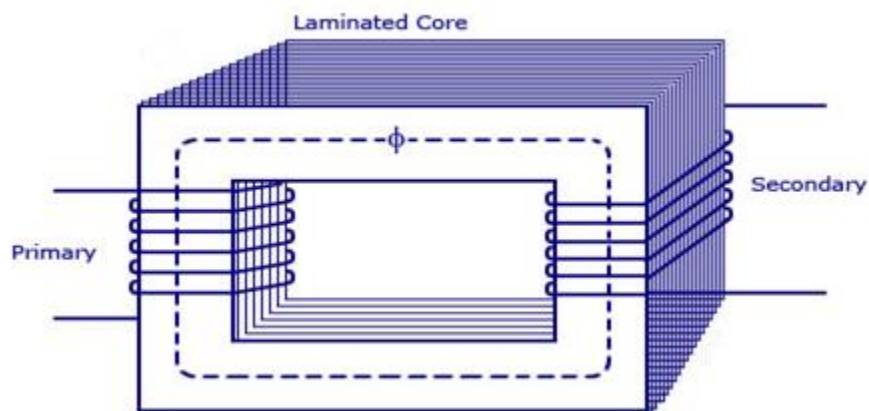
1. +12V – GND
2. -12V – GND
3. +5V – GND
4. -5V – GND

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.

2.2: 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:

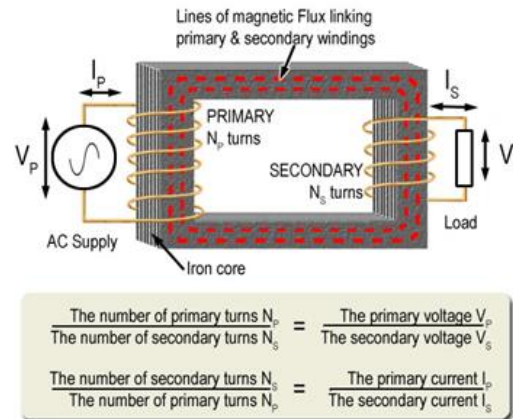
Figure 22, Construction of a power transformer



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:

Figure 23, Transformer turns ratio



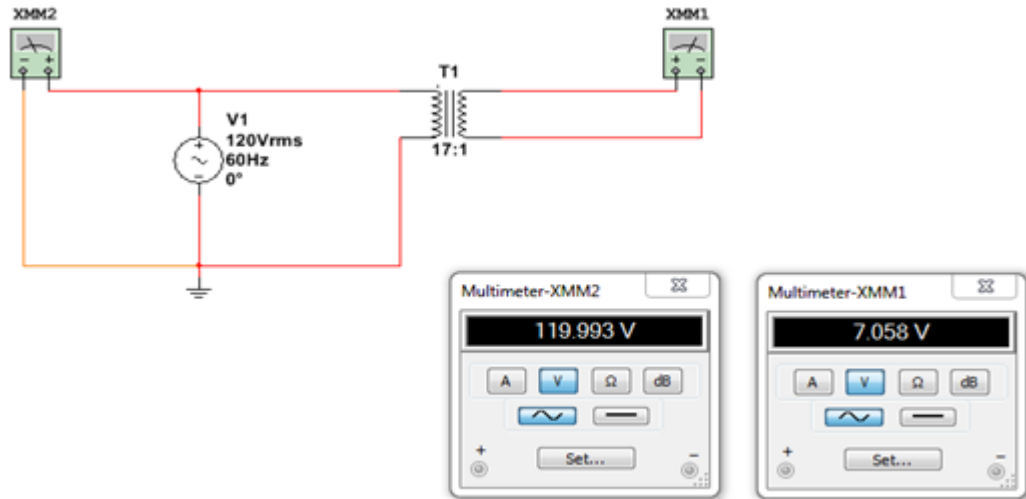
For the preliminary design, we assume that the maximum output current of our power supply is 2A. Hence, the output power is 10W for 5VDC and 6W for 3.3VDC output ports.

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.

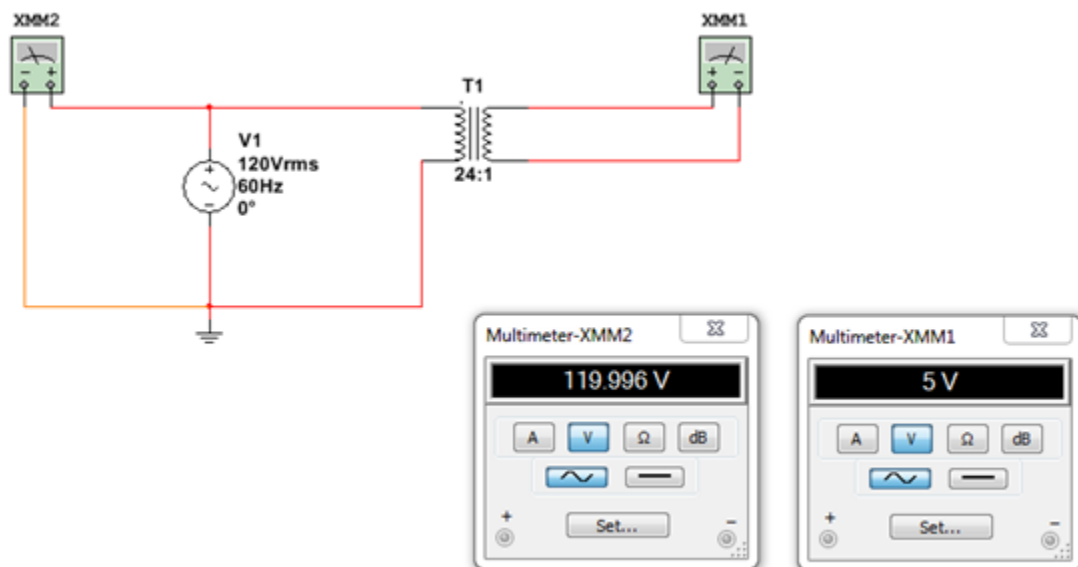
We are now going to use a simulation program to verify the primary and secondary voltages of the transformers. We are going to use MULTISIM simulation software in order to simulate the power supply circuit. In this section, only the transformer voltages will be determined using the transformer.

Figure 24, 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 25, Transformer design for 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.

2.3: Over-current Protection (Fuse)

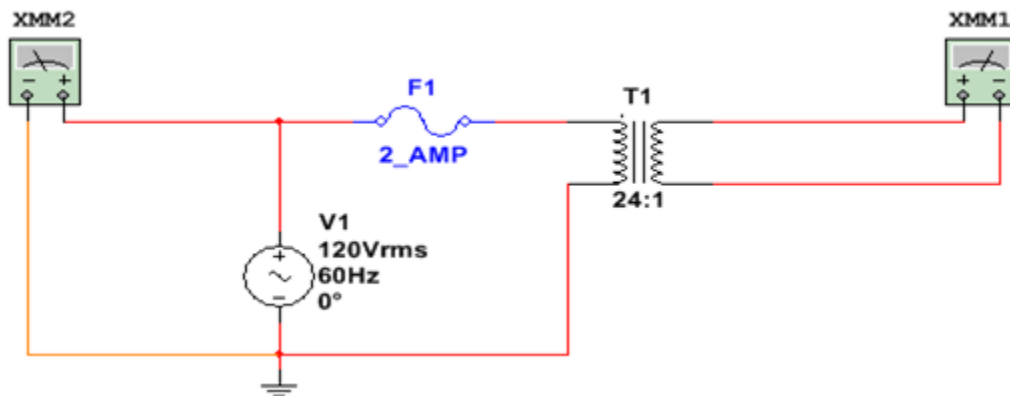
The nominal RMS voltage in North America is 120VAC however due to faults in power systems, this voltage can fluctuate significantly. The fault currents and voltages can easily damage sensitive electronic devices. Hence, it is important to provide safety features in the design of power supply. Currents 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 26, 2A fuse device for over current protection



The fuse is included in the following way in our power supply circuit.

Figure 27, 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.

2.4: Over-Voltage Protection (Varistor)

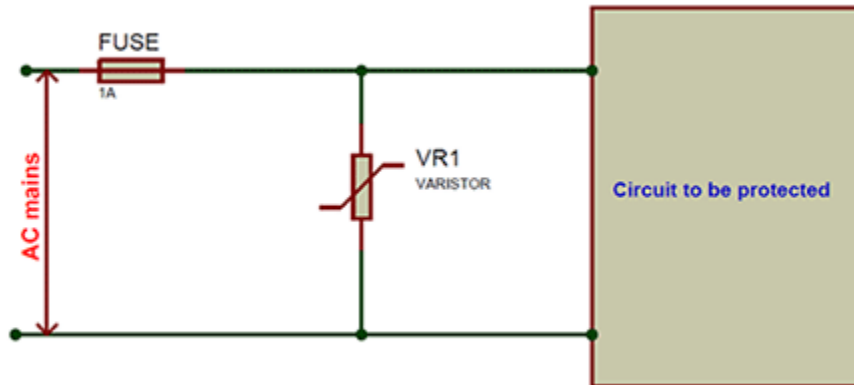
In the previous section, the operation of the fuse is explained in context of over current protection. However, current protection alone is not sufficient. Transient voltage surges and voltage swells can also damage the components. Therefore, it is also important to include over-voltage protection in 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 28, Metal oxide varistor



Figure 29, Placement of fuse and varistor in the circuit

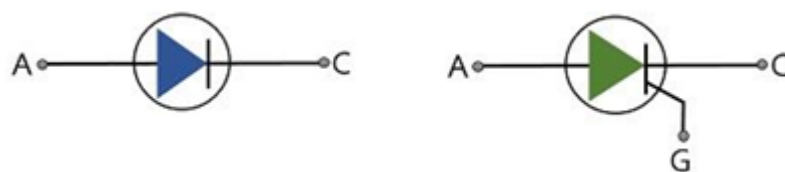


Using a combination of fuse and varistor at the input side of the power supply, the hazard of over current and over voltage can be reduced. These devices protect our circuit against high voltage transients and fault currents which can cause damage to the electronic components. These devices provide an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

2.5: 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 30, Diode and thyristor symbol

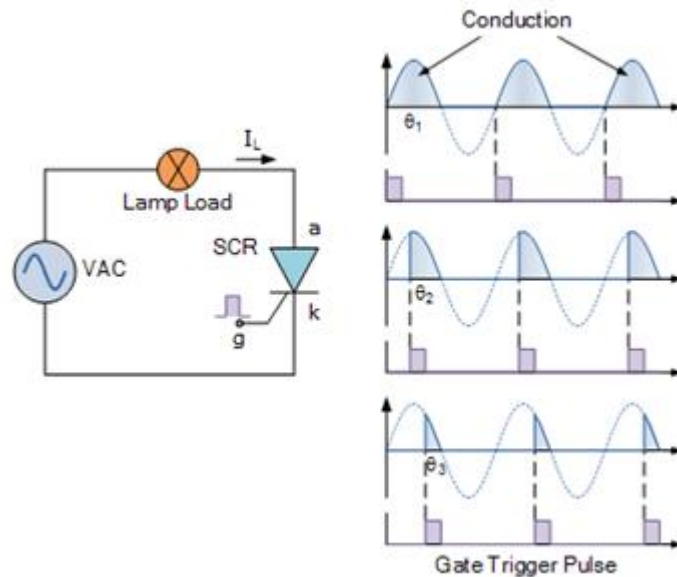


2.5.1: Silicon Controlled Rectifier

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 31, 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.

2.5.2: Diode Rectifiers

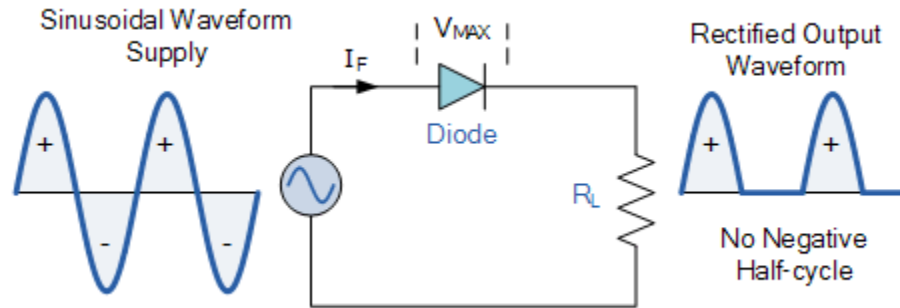
We have established the fact that diode rectifiers are simpler and cheaper in construction therefore we will be using diodes for the construction of rectifiers for our power supply. However, there are many different types of rectifiers and we need to choose a suitable rectifier type for our power supply. 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 32, Half wave diode rectifier



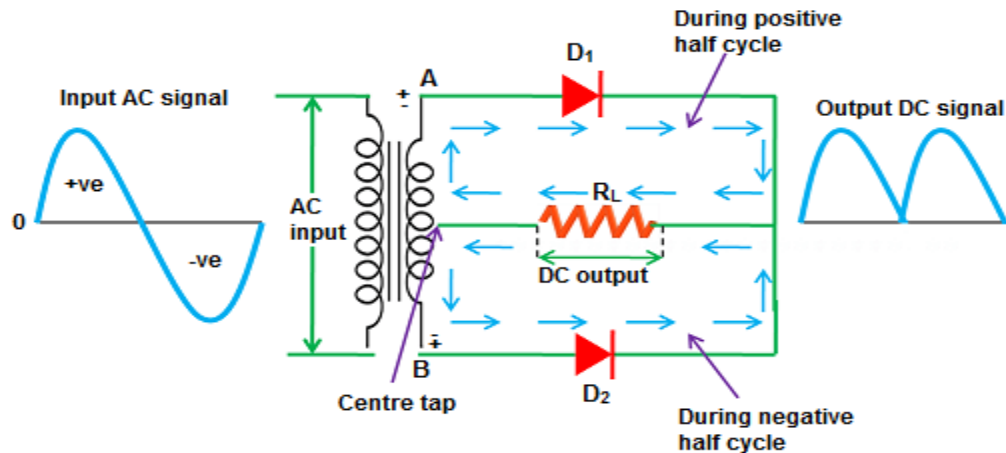
The diode is a unidirectional device which allows current in only one direction. Hence in the above circuit, the diode allows current flow only during the positive half cycle of the input voltage. During the positive half cycle, the anode voltage is higher than cathode voltage and thus the diode conducts 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.

This equation 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 33, Center tapped full wave rectifier

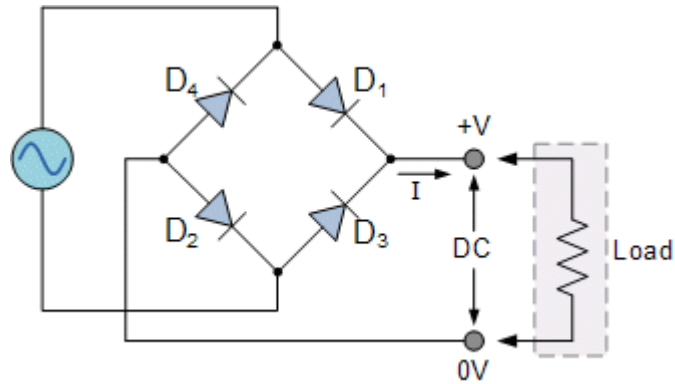


The center tapped rectifier requires a center tapped transformer. This configuration uses two diodes and therefore is more cost effective as compared to the full bridge rectifier which makes use of four diodes. However, the transformer required for this rectifier is heavier and more expensive. The transformer 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. Due to this limitation, this rectifier is not suitable for high power applications.

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:

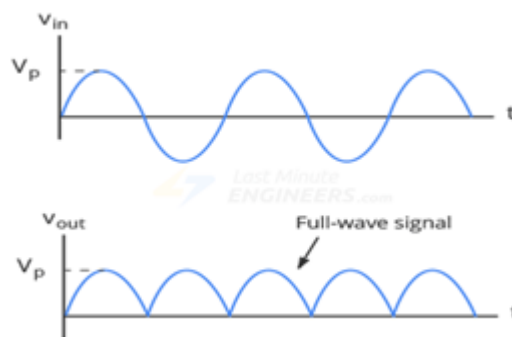
Figure 34, Full bridge rectifier



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:

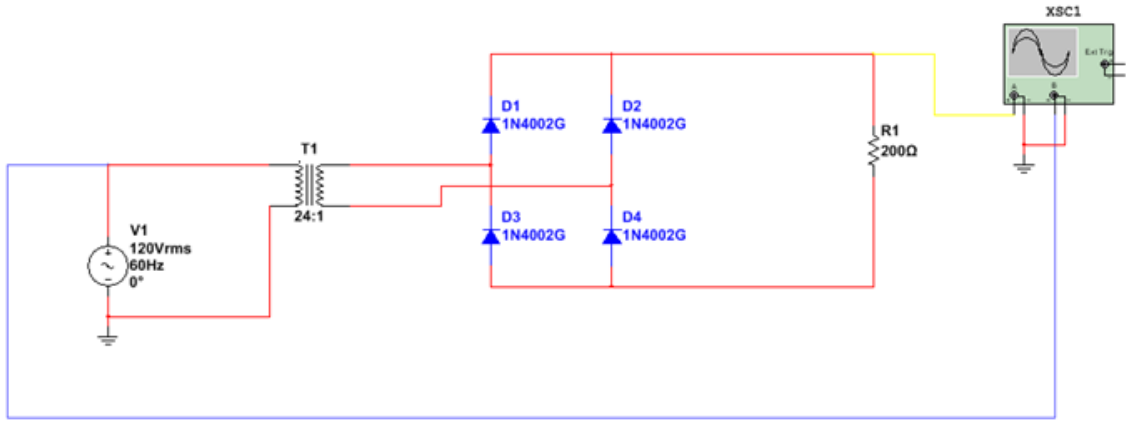
The input and output signal waveforms of the full bridge rectifier are presented in the following:

Figure 35, Input and output waveforms of the full bridge rectifier



The circuit schematic created in the MULTISIM simulation software is presented as following:

Figure 36, MULTISIM schematic of full bridge diode rectifier



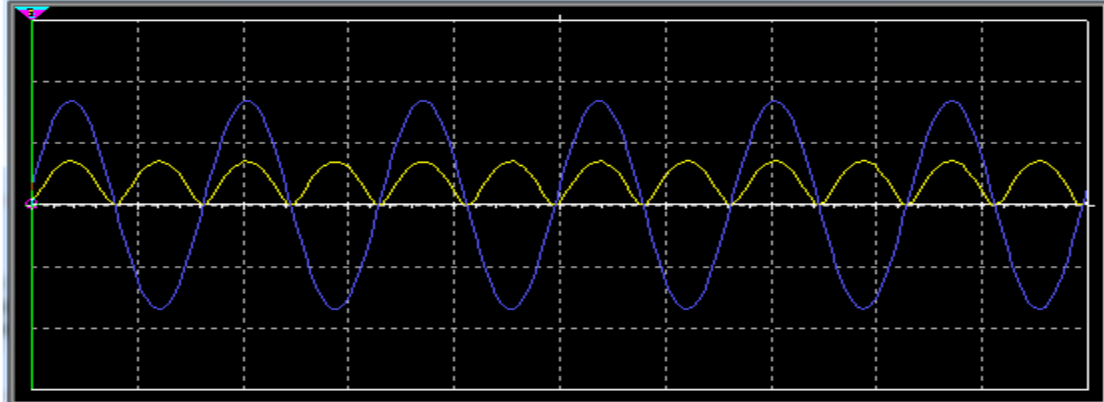
For an input voltage of 120VAC, the output voltage provided by this rectifier is 5VDC. The voltage at this level is obtained when the transformer turn ratio is 24:1 and load resistance is 200Ω.

Figure 37, 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 38, Input and output waveforms of the full bridge diode rectifier



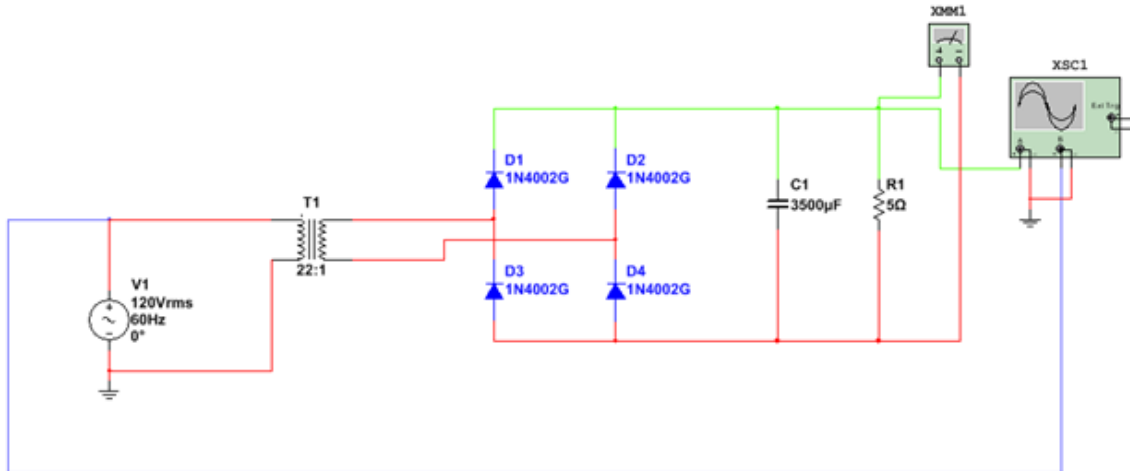
From this figure we can see that both the half cycles of the input voltage are reflected in the output. Therefore no power is wasted by this rectifier. However, there are still two main problems with this rectifier.

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.

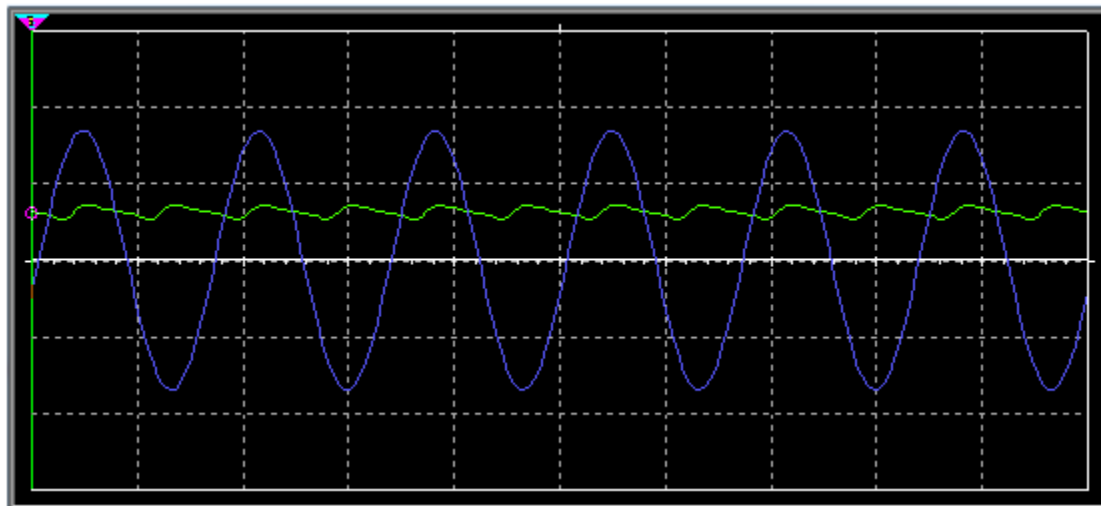
The second problem is that of ripple. The performance of this rectifier is better than half wave and center tapped rectifiers however the ripple still exists. In order to reduce the ripple, a filter capacitor needs to be used in parallel to the load. The selection of capacitor value is often done through trial and error. Therefore a few trials 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 39, Full wave diode bridge rectifier with 3500uF filter capacitor



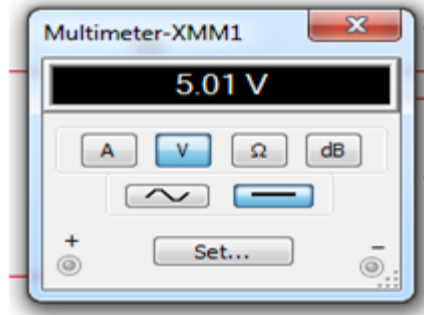
We notice that the value of the load resistor is changed to 5Ω . This value is selected because the required output current of the power supply is 1A. When the output voltage is 5V and load resistance is 5Ω then an output current of 1A is obtained. Hence, the output power of our designed power supply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

Figure 40, 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 41, Output voltage of the rectifier with 22:1 turns ratio and 3500uF capacitor

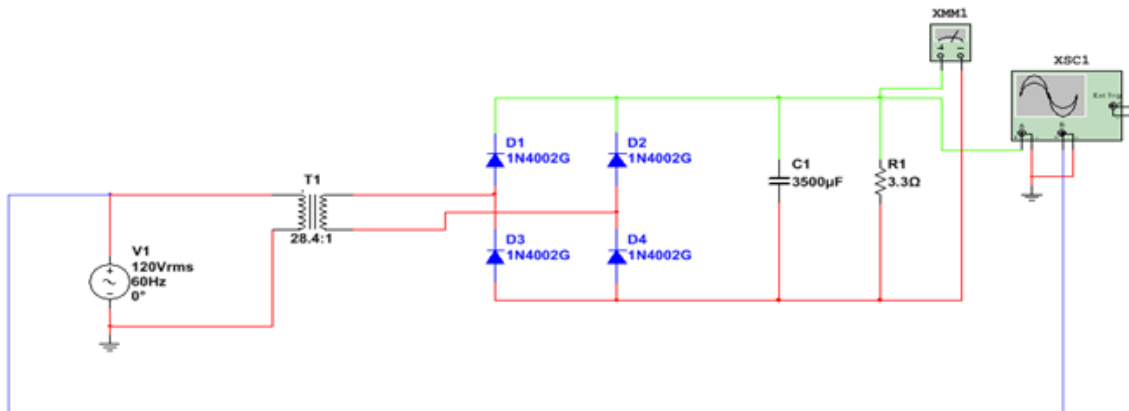


Since the required output voltage of 5VDC is obtained with an output current of 1A, we can say that the design objectives have been met. Hence, the filter capacitor of 3500 μ F will be used for our power supply circuit of 5VDC.

Full Wave Bridge Rectifier (3.3VDC)

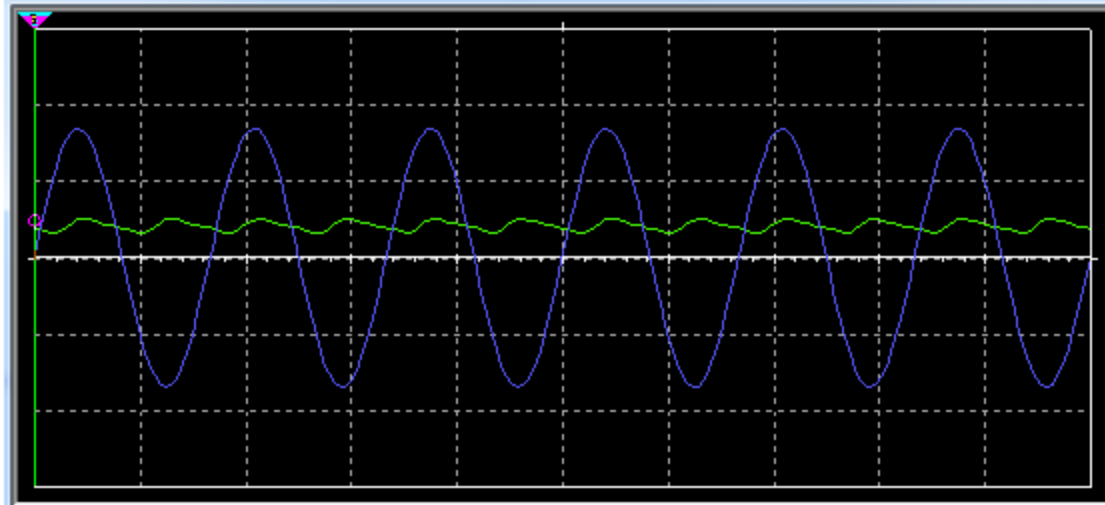
In the previous section, we have designed a full wave diode bridge rectifier circuit with an output voltage of 5VDC. However, according to the design requirements, the power supply must provide the provision for additional 3.3VDC. Therefore a separate rectifier circuit needs to be built for the 3.3VDC 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 3.3 Ω is needed to obtain an output current of 1A. The circuit schematic is shown as following:

Figure 42, Power supply circuit with 3.3VDC / 1A output



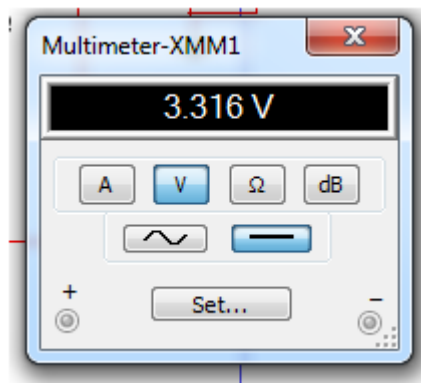
We notice that the turn ratio of the transformer is set to 28.4 whereas the value of the filter capacitor is kept the same. For this circuit, the input and output voltage waveforms are shown as following:

Figure 43, 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 44, 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.

2.4: 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 45, LM7805 voltage regulator

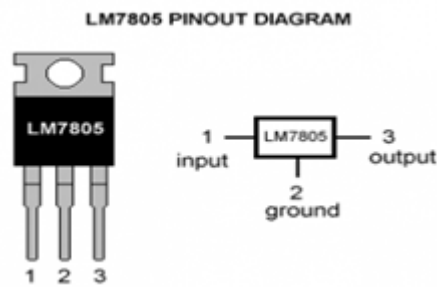
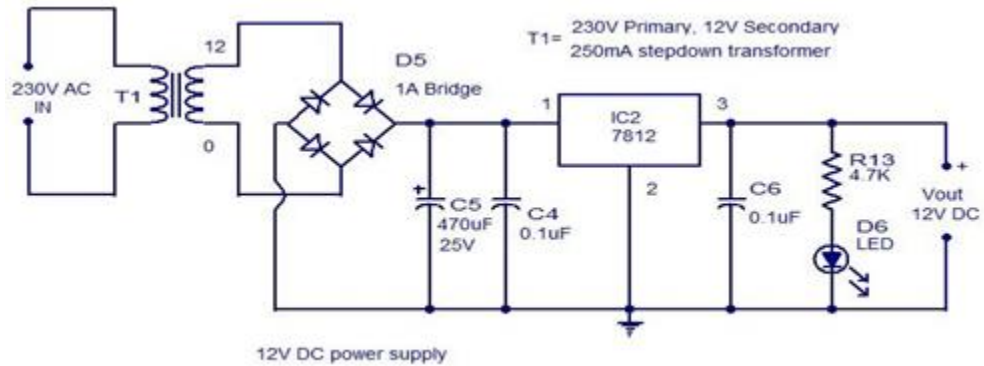
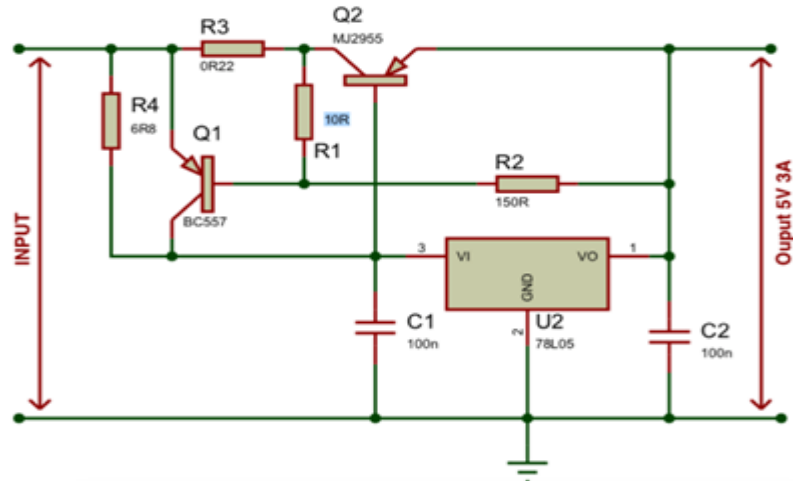


Figure 46, Example of a voltage regulator based AC-DC power supply



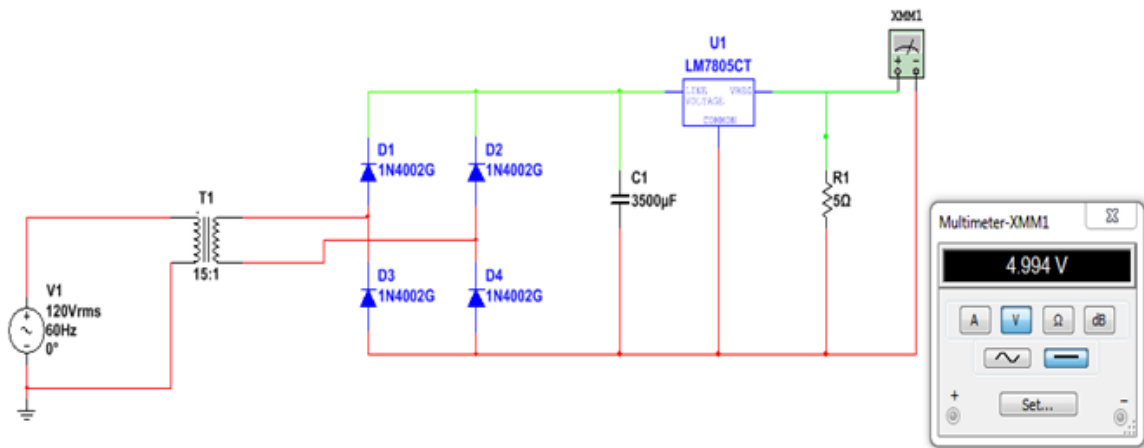
The internal circuitry of the voltage regulator IC is shown as following:

Figure 47, Internal circuit of voltage regulator IC



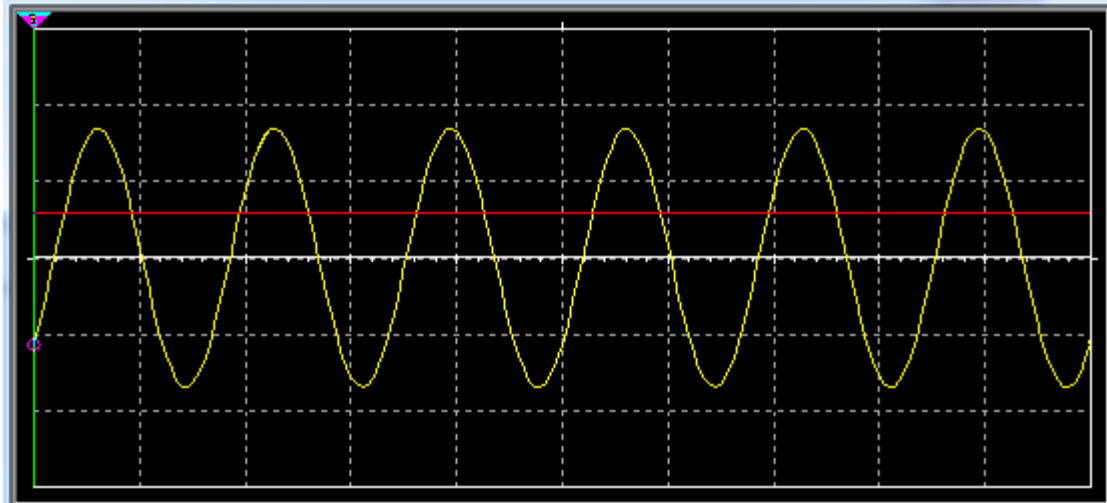
Now we will place the 7805 voltage regulator in our 5VDC sub-circuit in order to get a stable 5VDC signal across the load resistance. The MULTISIM simulation circuit with the voltage regulator is provided as follow:

Figure 48, 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 49, 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.

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 add-on. 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 50, An Arduino Uno board

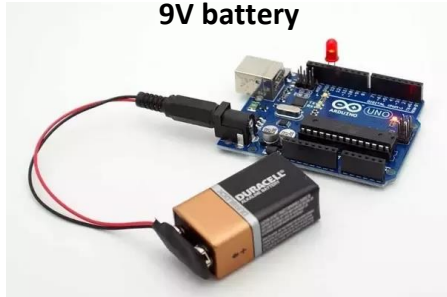


The first microcontroller is the Arduino Uno. It uses an ATmega328 microchip for its CPU. This 80bit RISC-based microcontroller gives 32KB ISP Flash memory that can read-while-write. It has 32 general registers, serial programmable USART (used to receive and transmit communication with a computer), a 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 51, Arduino Uno powered by 9V battery

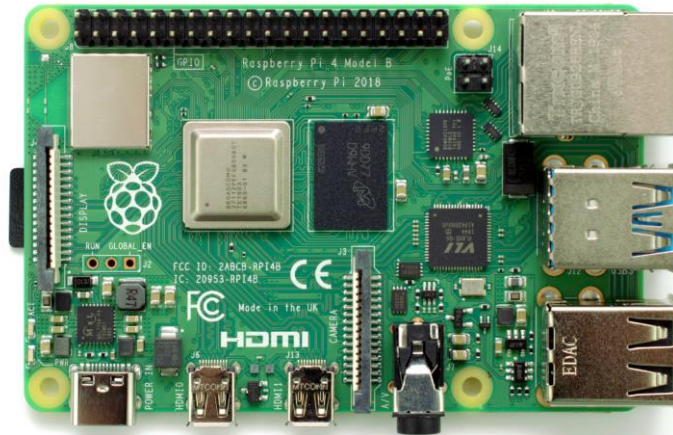


The Arduino Uno is priced at around \$22, Wi-Fi shield less than \$10, and sensors all coming together for around \$20 for a total of \$60 with any other fees. Many starter kits also are sold for around \$50 or \$60, that include all of the wires and any additional components such as breadboards or resistors.

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.

Raspberry Pi Microcontroller

Figure 52, 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.

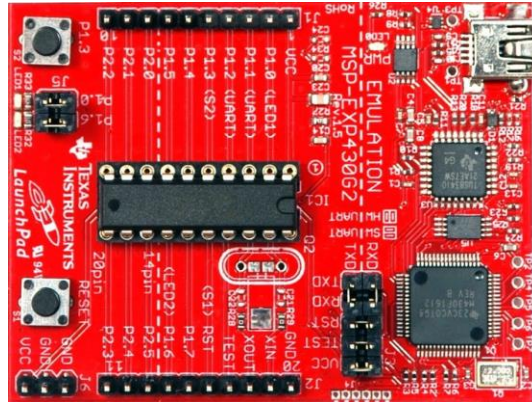
Using power is not the only problem the Raspberry Pi has, giving it power has limited options. The Arduino Uno could be powered with a battery, outlet, or microUSB/USB-C. The Raspberry Pi can only be powered with a microUSB/USB-C cord, meaning it needs an external USB power bank or another computer.

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.

MSP430 Microcontroller

Figure 53, 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.

Microcontroller Pros/Cons

Table 3, Pros and Cons of Arduino Uno, Raspberry Pi, and MSP430

Microcontroller	Pros	Cons
Arduino Uno	<ul style="list-style-type: none"> • Can use a battery pack for power • Variety of electrical components • Many libraries to hand electrical components • Open source • Parts are low cost • Has 32KB memory • Output max of 5V • 20 IO pins 	<ul style="list-style-type: none"> • Cannot connect to the internet on its own • Does not have its own power source • Only 1 USB port • Microcontroller fairly pricey, around \$25 • Coded in C++, not a main language we learn
Raspberry Pi	<ul style="list-style-type: none"> • Can run multiple programs at once • 4 separate USB ports • Can be coded in 4 different languages • Supports its own touchscreen display or HDMI display • Has RAM slot up to 8GB • Programmable within itself 	<ul style="list-style-type: none"> • Does not have its own memory • May crash if it has corrupt files or problems • Drains power very quickly • Can only be powered with USB connections
MSP430	<ul style="list-style-type: none"> • Very cheap • Low power consumption • Can be used in other IDEs such as Code Composer Studio 	<ul style="list-style-type: none"> • Only outputs 3.6V • Only 16 IO pins

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.

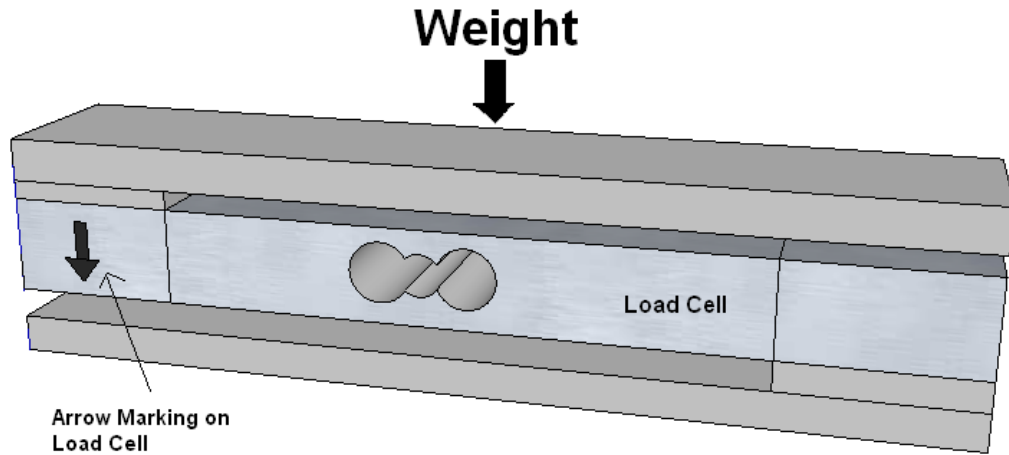
Sensors

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.

HX711 Load Cell

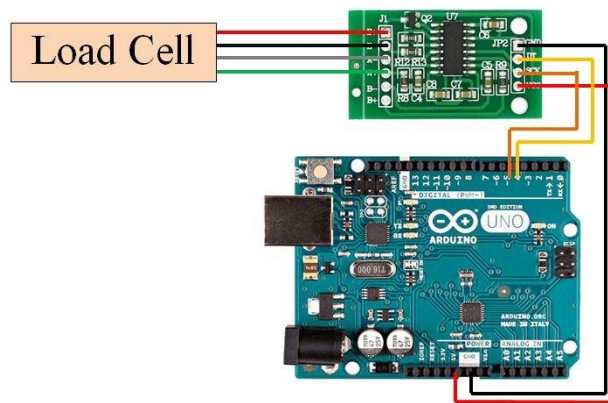
Figure 54, A Load Cell diagram



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.

Programming the HX711 to the Arduino is simple. Using the HX711 library and the correct layout, we can easily calibrate them together. A sample diagram provided by electopeak is shown.

Figure 55, 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.

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. The signal will travel to the Arduino to light up the respective LEDs.

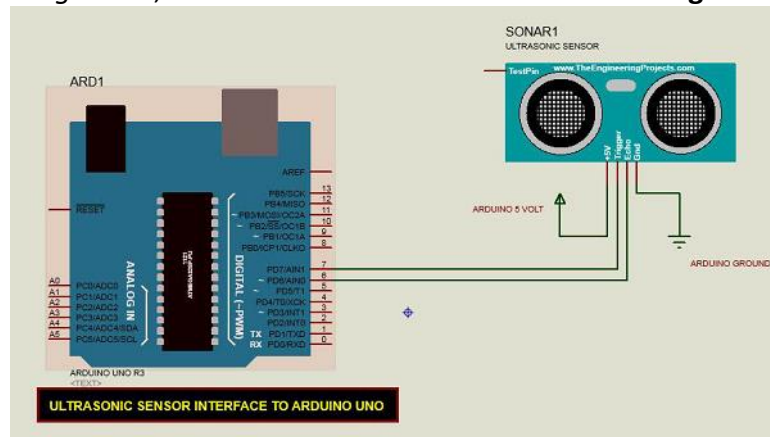
HC-SR04 ultrasonic sensor

Figure 56, 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 57, Arduino Uno connected to HC-SR04 diagram



Wifi and Mobile

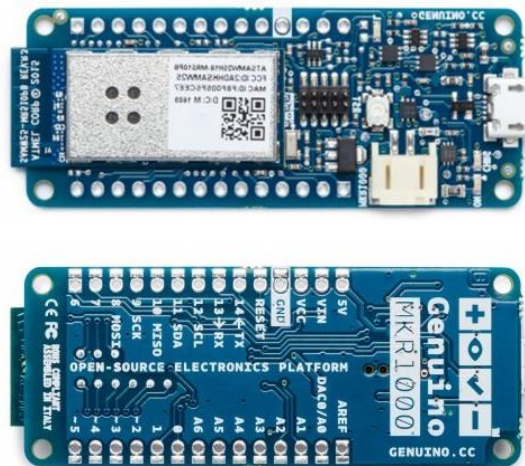
One of the motives of our product is that many people have pets but also have a busy schedule. Many times we may leave our homes and forget to feed our pets, or are gone for long periods of time and feel the need to rush home just to feed them. So one of the ways we want to make this product fix that solution is to have a mobile app that can manual dispense food for our pets, or have the ability to set a timer remotely that will dispense food during the schedule eating time that the owner arranges.

To be able to do this we would need to be able to connect to the pet feeder via wifi so there is enough range if we are not home. Also we would need a mobile app to send controls and information to the pet feeder so it can set timers or dispense food. Below are a few wifi modules and options that are compatible with the MCU that we have selected, which is the Arduino Uno, and will give our product the ability to allow users to connect to it wirelessly from their mobile device.

Wifi chip comparisons:

MKR1000:

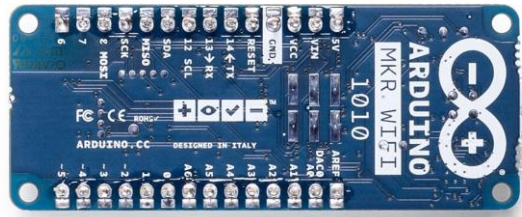
Figure 58, MKR1000 Wifi Module



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.

MKR1010:

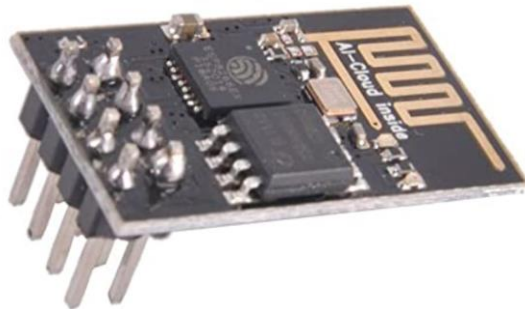
Figure 59, MKR1010 Wifi Module



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.

ESP8266:

Figure 60, 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.

ESP32:

Figure 61, 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.

Arduino uno wifi Rev 2:

Figure 62, Arduino wifi Rev 2 MCU with integrated wifi



This Board is very different from the previous 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.

Table 4, Pros and Cons of Wifi modules

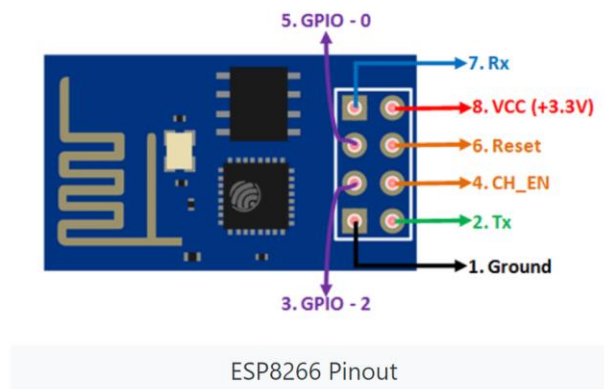
Boards	Pros	Cons
MKR1000	<ul style="list-style-type: none"> • Good for individuals new to wifi development • Has many libraries to help with wifi development 	<ul style="list-style-type: none"> • More Expensive • Sensitive I/O pins • No bluetooth (optional) • Possible soldering needed
MKR1010	<ul style="list-style-type: none"> • Good tech documentation and support • Bluetooth • Easy to use 	<ul style="list-style-type: none"> • More Expensive • Possible soldering needed
ESP8266	<ul style="list-style-type: none"> • Cheap • Easy wifi development • Lots of existing libraries for wifi development 	<ul style="list-style-type: none"> • No bluetooth (optional) • Not robust when it comes to upgrades • Possible soldering needed
ESP32	<ul style="list-style-type: none"> • Mid-cost • Bluetooth • Robust when it comes to upgrades 	<ul style="list-style-type: none"> • Not all libraries compatible with arduino • Less support • Possible soldering needed
Arduino uno wifi Rev 2	<ul style="list-style-type: none"> • Integrated wifi and bluetooth • One board no need for soldering 	<ul style="list-style-type: none"> • More Expensive • May have missing libraries for wifi development

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 63, 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:

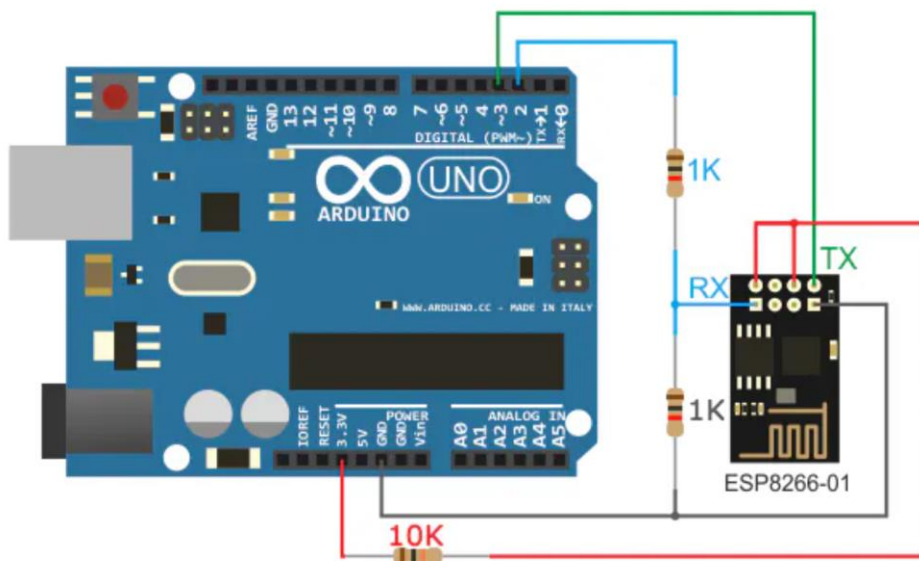
- Low cost, compact and powerful Wi-Fi Module
- Power Supply: +3.3V only
- Current Consumption: 100mA
- I/O Voltage: 3.6V (max)
- I/O source current: 12mA (max)
- Built-in low power 32-bit MCU @ 80MHz
- 512kB Flash Memory
- Can be used as station or Access Point or both
- Supports Deep sleep (<10uA)
- 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 64, 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.

Figure 65, Wifi setup code snippet

```

#include <ESP8266WiFi.h>

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

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

```

From the image above we can see that we will need to import the “ESP8266WiFi.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 66, AT Command

Commands	Description	Type
AT+RST	restart module	basic
AT+CWMODE	wifi mode	wifi
AT+CWJAP	join AP	wifi
AT+CWLAP	list AP	wifi
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 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 ESP8266’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 68, Void Loop code snippet for main functions

Figure 67, TCP Client

The screenshot shows a mobile application interface titled "Demo" with the subtitle "Add a item". It contains three input fields: "Name" with the text "Test", "IP" with the text "192.168.4.1", and "Port" with the text "80". Below the input fields are two buttons: "Add" and "Cancel".

```
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.

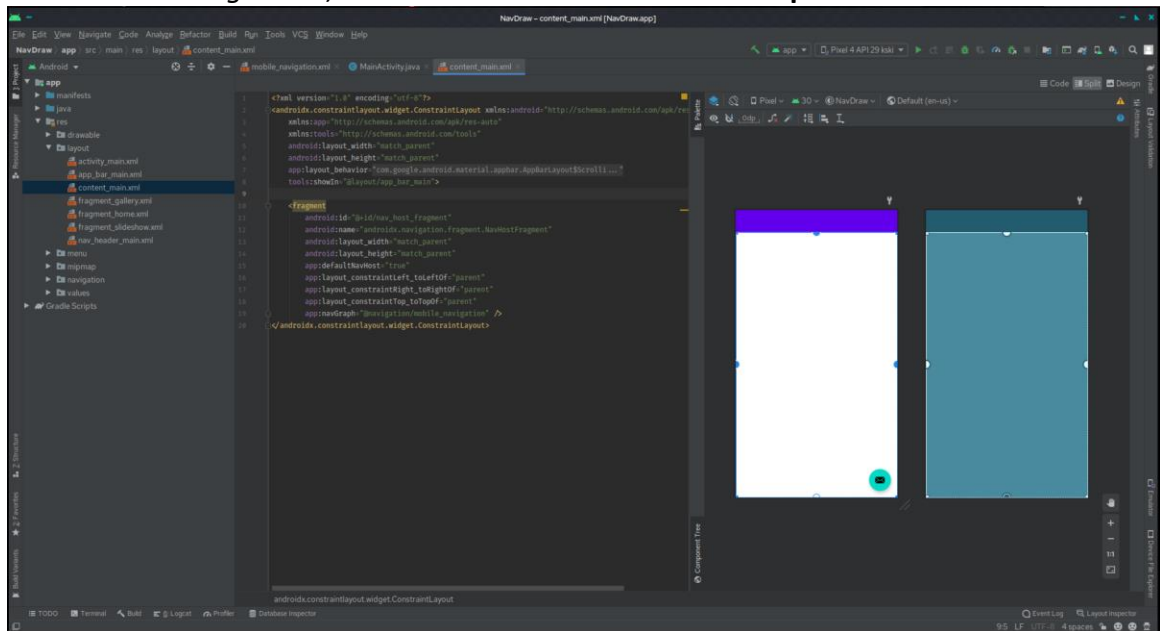
Mobile APP Environment:

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.

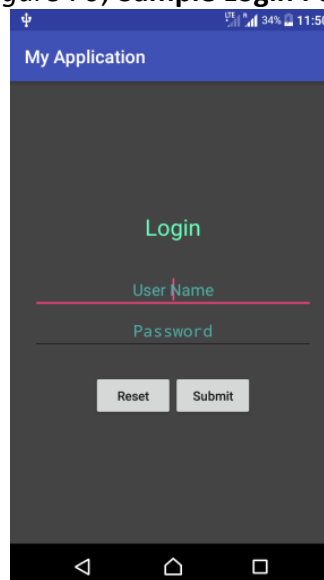
Figure 69, Android Studio code and visual representation



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 70, Sample Login Page

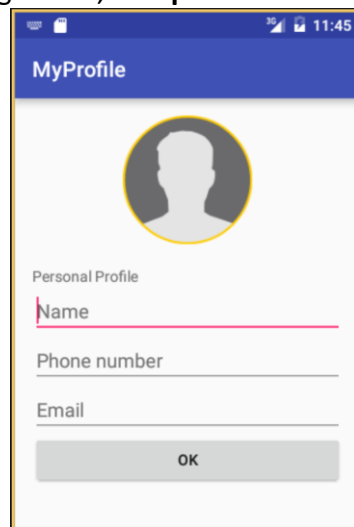


The first thing that we would like to have on the mobile app is a login screen that allows only users that we have registered being able to use the app, this way it would be easier

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 71, 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 on 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.