AUTOMATED PET FOOD DISPENSER

By

Kevin Shibley

Senior Project

ELECTRICAL ENGINEERING DEPARTMENT

California Polytechnic State University

San Luis Obispo

June 2014

Table of Contents

LIST OF FIGURES

LIST OF TABLES

Acknowledgements

- Sue Shibley – mother and inspiration for this project.
- David Shibley – brother, friend, and co-inventor
- Dr. Bridget Benson – advisor and inspirational professor
- Electrical Engineering Department at Cal Poly, San Luis Obispo

Abstract

Electronics revolutionize the world and simplify life. The Automated Pet Food Dispenser (APFD) revolutionizes the pet industry and provides unprecedented food capacity and delivery capability, giving pet owners worry-free vacationing. Sensors and timers optimize functionality and reduce waste while improving reliability and usability. User-defined settings and customization gives the device adaptability to meet every customer's needs.

I. Introduction

This report considers the current marketable pet food dispenser models and suggests an improved design. The context and motivation for the project is discussed and background information is given before requirements and specifications are considered. Finally, the design and testing of the product is covered.

A. Context

Existing pet food dispensers provide minimal adaptability or user personalization. Figures 1 and 2 show the most common models. Despite their popularity, both units lack functionality and have critical design flaws. The design in Figure 1 continuously provides food by utilizing gravity. This results in overfeeding and, consequently, premature food depletion. The design in Figure 2 prevents overfeeding by restricting the quantity dispensed per time period but lacks any animal feedback. This risks waste and lacks adaptability. Both designs are small and limit storage capacity.

Figure 1: Gravity Actuated Dispenser

Figure 2: Timer Actuated Dispenser

B. Motivation

Pets can severely limit their owner's freedom by requiring life's basic necessities: food, water, and shelter. While the latter two are often a non-issue, ensuring they are properly fed remains a constant headache for the average owner. Complications increase when pet owners take vacations and leave their animals behind. The APFD removes this inconvenience by regulating the quantity and frequency of food delivered while providing user feedback and customization.

II. Background

The Automated Pet Food Dispenser uses common electronic components including an ultrasonic ranging module, LCD screen, Servo Motor, and the Atmega328P chip on an Arduino board. The datasheets for each of these components can be found in [1]-[4].

The HC-SR04 Ultrasonic Ranging Module detects an object's range by sending and receiving a 40kHz signal. The time required for the signal to return determines the distance by Eq. 1.

$$range = (t_{high} * v_{sound})/2 \qquad \textit{Equation 1}$$

The servo motor operates on a pulse width modulated (PWM) signal. It rotates between 0° and 180° according to the duration of the pulse. A 1700µs pulse will center the platform at 90° while increasing and decreasing the pulse width increases and decreases the angle, respectively. A signal should be sent every 20ms to function properly.

III. Requirements

In developing the APFD, many factors and customer needs were considered which defined the requirements for the project.

1. At minimum, this device must deliver food to an animal automatically based on two primary factors: time and proximity. In practice, the device detects when the animal is near before delivering food and waits for a specified time before delivering food again.
2. The option to customize the duration of time between feedings.
3. The option to customize the amount of food delivered
4. The ability to display the following pertinent information:
    a. Food remaining in the container
    b. Time remaining until the next feeding
    c. The ability to service multiple pets with varying diets. (All except for the last of these functions were incorporated in the final prototype of the APFD.)
5. Finally, the product must be easily powered.

IV. Specifications

A. Structure

The structure of the APFD should be 1’ wide by 1’ deep by 4’ tall to be compact while allowing easy, waist-high access to users to refill with pet food. There must be a base upon which the food is dispensed through the feeding system.

B. Feeding System

The feeding system consists of a food container, funnel, and a feeder door. The food container must be 12” in diameter and hold up to 30lb of dog food. The funnel must be 12” in diameter at the top and taper down to 2” in diameter at the bottom. This is the ideal diameter for controlled flow of large dog food; diameters smaller than 2” will create blockage. The feeder door must support up to 30lb of pressure while closed and be able to open and close within 0.5 seconds. Longer open and close times create unpredictable amounts of food dispensed. The feeder door is

operated by a servo motor which has 180° of rotation, operates on 5V, and receives its signal from an Atmega328P microcontroller.

Upon detection of the animal, the system must be capable of dispensing food in single cup increments. Time between feedings must also be adjustable to one hour increments with a 1 second tolerance.

C. Detection

The detection system consists of a 40kHz ultrasonic ranging module that detects objects up to 4.5’ away at an angle of 45° from the top of the unit. This results in a trip when an animal is within 2’ of the device to minimize nuisance tripping while allowing for practical operation. The ranging module must operate on 5V and communicate to the microcontroller through a Trigger and Echo pin.

D. Display

The display should be a backlit LCD screen running on 5V and capable of displaying 2 lines of 5x8 dot segments with 16 segments per line. The LCD screen must operate in nibble mode. The display has three screen options: a main menu, an information screen, and a confirmation screen.

The main menu has four components: quantity selection, frequency selection, enter, and information. The quantity selection allows the user to increase or decrease the number of cups dispensed. Similarly, the frequency selection allows the user to increase or decrease the number of hours between feeding. Enter enables the confirmation screen which informs the user that the new values have been entered. Information enables the information screen which informs the user of the time remaining until the next feeding and the amount of food remaining in the container.

E. Display Control

The display is controlled by four 1000mm^3 buttons: up, down, left, and right. These four buttons only function in the main menu. The number of cups and hours can be increased or decreased when selected using the up and down buttons. The up and down buttons also are used to activate

the confirmation screen or the information screen when enter or information is selected, respectively. Left and right buttons cycle through the four menu options.

F. Microcontroller

The microcontroller used is an Atmega328P. It must have at least 12 I/O ports plus a 5V rail and ground. There must be a 16MHz crystal oscillator used as a clock and at least 3 timers available, one of which must be a 16bit timer. The 16bit timer is necessary to accurately measure seconds, minutes, and hours for the feeder timer. One 8bit timer is used to generate a PWM signal that drives the servo motor. The other 8bit timer is reserved for use by the ultrasonic ranging module.

V. Design

The APFD underwent three design cycles. Phase 1 was a rudimentary proof of concept while Phase 2 incorporated more useful functions. Phase 3 added additional features and fine-tuned existing ones. Expectedly, each phase involved both hardware and software redesigns, though mechanical designs remained unchanged through each phase.

A. Mechanical

Separate from the hardware and software, the mechanical design was done first since it formed hardware (and by extension, software) constraints. There were three main components to the design: the food container, the food funnel, and the feeder door.

The initial design for the food container is shown in Figure 3. The concept behind this design was collapsibility to maximize space efficiency when less food was needed. This design was never able to be implemented, however, due to limited supplies and construction capabilities. Instead, a 5 gallon water bottle was used because of its durability and accessibility. As shown in Figure 4, the water bottle functions upside down with the bottom cut off for food access.

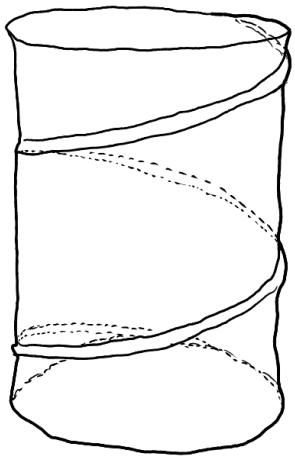

Figure 3: Food container – first design

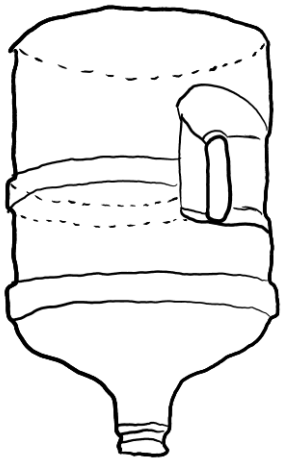

Figure 4: Food container – final design

Selecting the water bottle as the container also satisfied the funnel requirements, tapering down from the width of the container to 2” in diameter (the ideal diameter for controlled food flow). Additionally, this funnel fit well in a 2” PVC pipe, allowing for further flexibility in the funnel’s design. As shown in Figure 5, PVC pipe delivers food down at an angle much more accessible to the animal.

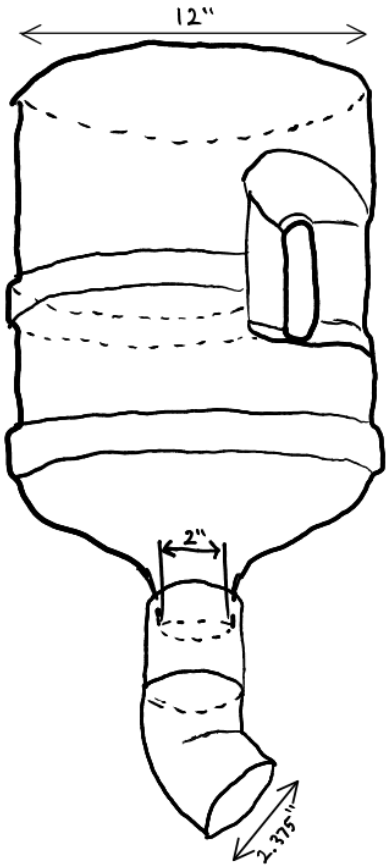

Figure 5: Food Funnel

Finally, two versions of the feeder door were designed before one was suitable for this application. The first attempt used a butterfly valve. The purpose of this design was to be sleek and compact, with no protruding parts. Unfortunately, this design clogged the funnel and prevented food from flowing through even while open. This design was then replaced with a moving platform as shown in Figure 6.

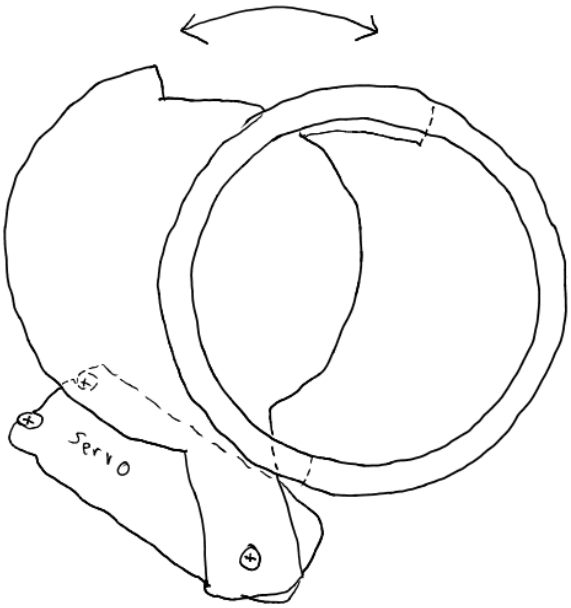

Figure 6: Feeder Door

The final mechanical design was relatively simple, consisting of a container, funnel and a feeder door, which blocks the funnel opening and is actuated by a servo motor. Finally, the container sits atop the cardboard base which provides height to the APFD. The materials used in the construction of this prototype consisted of cardboard, duct tape, a servo motor, PVC piping, and the top of a 5 gallon water bottle.

B. Phase 1

The first phase of the design cycle proved the concept of combining time and proximity to create a better pet feeder. The level 0 block diagram in Figure 7 shows the simplistic initial design with only two inputs and one output. The level 1 block diagram in Figure 8 shows how the two inputs result in the single output, where solid lines indicate feeder piping and dotted lines indicate electrical signals.

Figure 7: APFD Level 0 Block Diagram – First Design

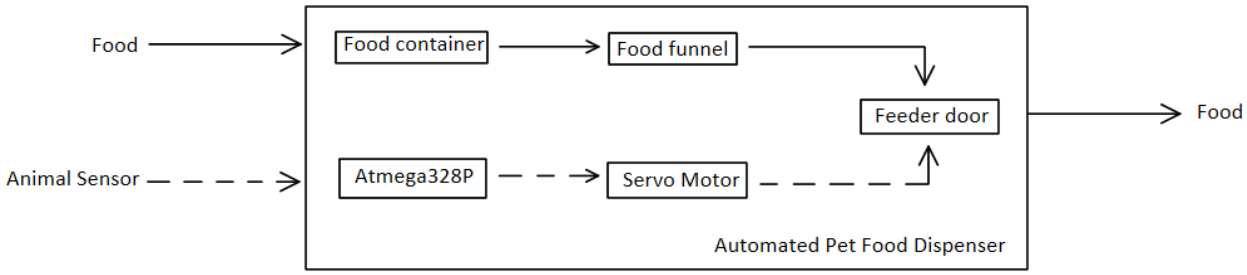

Figure 8: APFD Level 1 Block Diagram – First Design

i. Hardware

The hardware involved in phase 1 consisted of an Arduino UNO with the Atmega328P, a 40kHz ultrasonic range finder, and a servo motor. In order to meet the detection design specifications, the range finder was mounted atop the container at a 45° angle towards the floor. The servo motor was mounted against the feeder funnel and the platform to the feeder door was attached to the arm of the servo motor, allowing the platform to swing in and out of the slot as depicted in Figure 6. Both the ultrasonic range finder and the servo were electrically connected to the Atmega328P as shown by the wiring diagram in Figure 9.

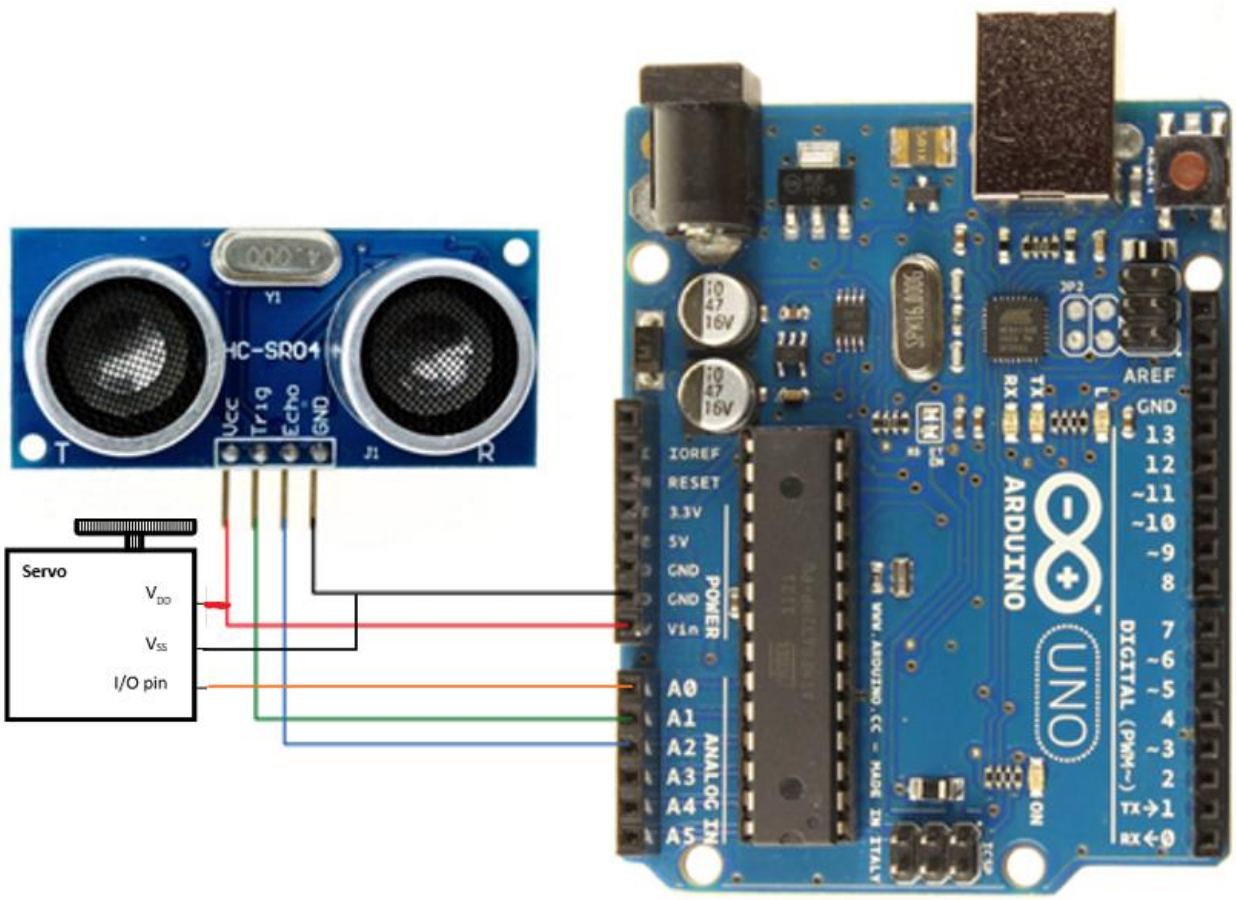

Figure 9: Phase 1 Wiring Diagram

ii. Software

The software program in phase 1 used three functions, an ISR, and one infinite loop in which everything takes place after a short initialization process. This process is depicted by the flow diagram in Figure 10.

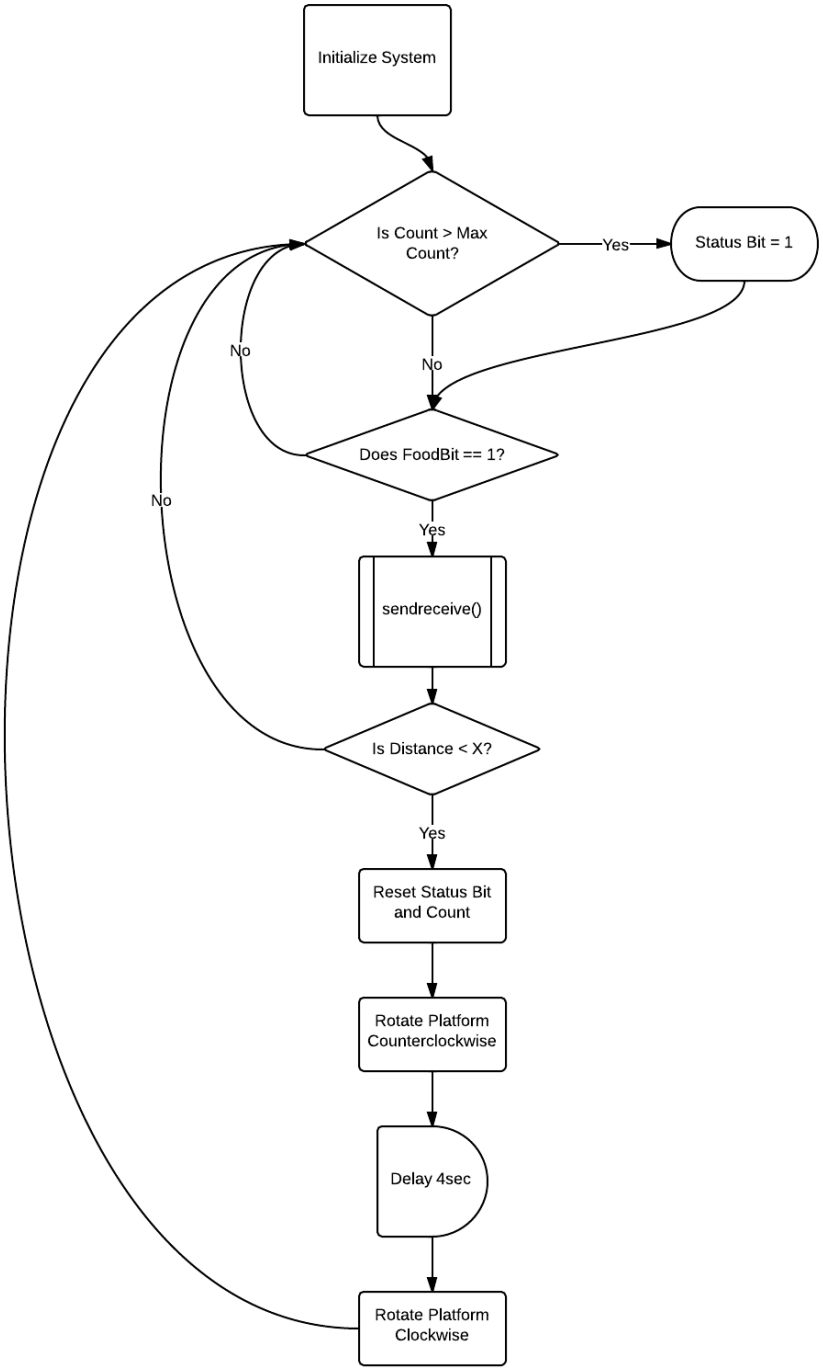

Figure 10: Phase 1 Flow Diagram

The initialization process sets the I/O pins functionality, initializes timers 1 and 0, and enables interrupts. PC0 and PC1 drive the signal to the servo motor and the trigger pin on the ultrasonic module, respectively, while PC2 is an input from the ultrasonic module echo pin. Timer0 is

initialized in its standard overflow mode with a prescaler of 64 using the initTimer0 function. This timer is used to limit the frequency of allowable food dispensation by counting seconds via an interrupt on a compare match. Timer1 is then initialized to overflow at max (0xFFFF) with a prescaler of 64. This timer is used to calculate the distance from an object by measuring the time required for the ultrasonic ping to return.

The three functions used are initTimer0, send_trigger, and send_receive. The initTimer0 function initializes Timer 0 to be used in CTC mode with a prescaler value of 64 and a TOP value of 0xF9 with interrupts enable upon a compare match. This produced an interrupt frequency of 16MHz / 64 / 249 = 1004 Hz to allow for accurate measurement of time elapsed since the last dispensing of food. This is done through the ISR by incrementing a counter, Counter, which is then checked and compared to a top value calibrated to equal one second. The send_trigger function sends the necessary pulse to the ultrasonic sensor to emit the ultrasonic pulse. The send_receive function begins with the send_trigger function then waits for the echo signal to drop to reset TCNT1 (the timer value of Timer1). When the echo signal returns high, the value of TCNT1 is returned.

The final section of the program is the infinite while loop. The loop starts by checking Count and setting a status bit FoodFlag if Count is greater than the time set between feedings (in this phase, for testing purposes, it was 10 seconds). FoodFlag is is then checked. If FoodFlag equals 0, the loop restarts. Otherwise, send_receive is assigned to a variable and is checked. If the value of the variable is greater than a certain distance, there is no animal present and the loop restarts. Otherwise, status_bit and myCounter is reset and the servo motor is rotated 120 degrees CCW to release food. After a few seconds of releasing food, the servo motor is rotated 120 degrees back CW to close the food opening and the loop restarts.

C. Phase 2

The second phase of the design cycle elaborated on the concept of combining time and proximity to create a better pet feeder by introducing user customization via directional buttons and an LCD screen. The level 0 block diagram in Figure 11 shows the slightly more involved second design with one additional input and output. The level 1 block diagram in Figure 12 shows how the new input results in a new output, where solid lines indicate feeder piping and dotted lines indicate electrical signals.

Figure 11: APFD Level 0 Block Diagram – Second Design

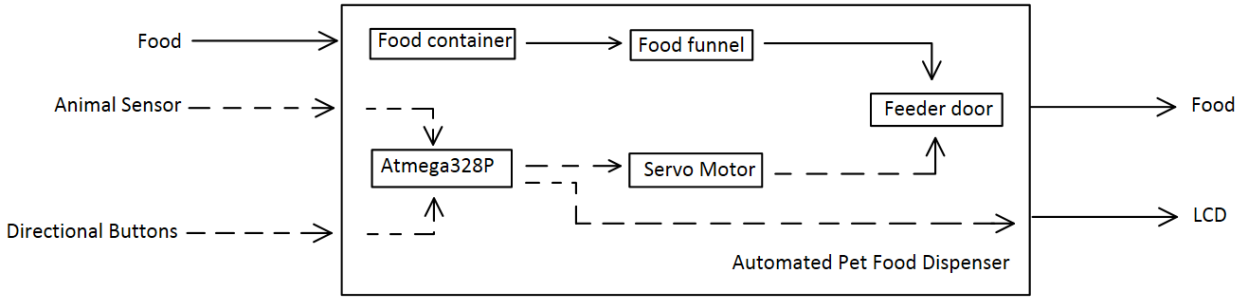

Figure 12: APFD Level 1 Block Diagram – Second Design

i. Hardware

In addition to the hardware components involved in phase 1, phase 2 included directional buttons and an LCD module. The directional buttons (up, down, left, and right) were used as inputs to the microcontroller for user input. The LCD module was used to display menu options to the user including pertinent information and the ability to adjust settings. The updated wiring diagram for phase 2 is shown in Figure 13.

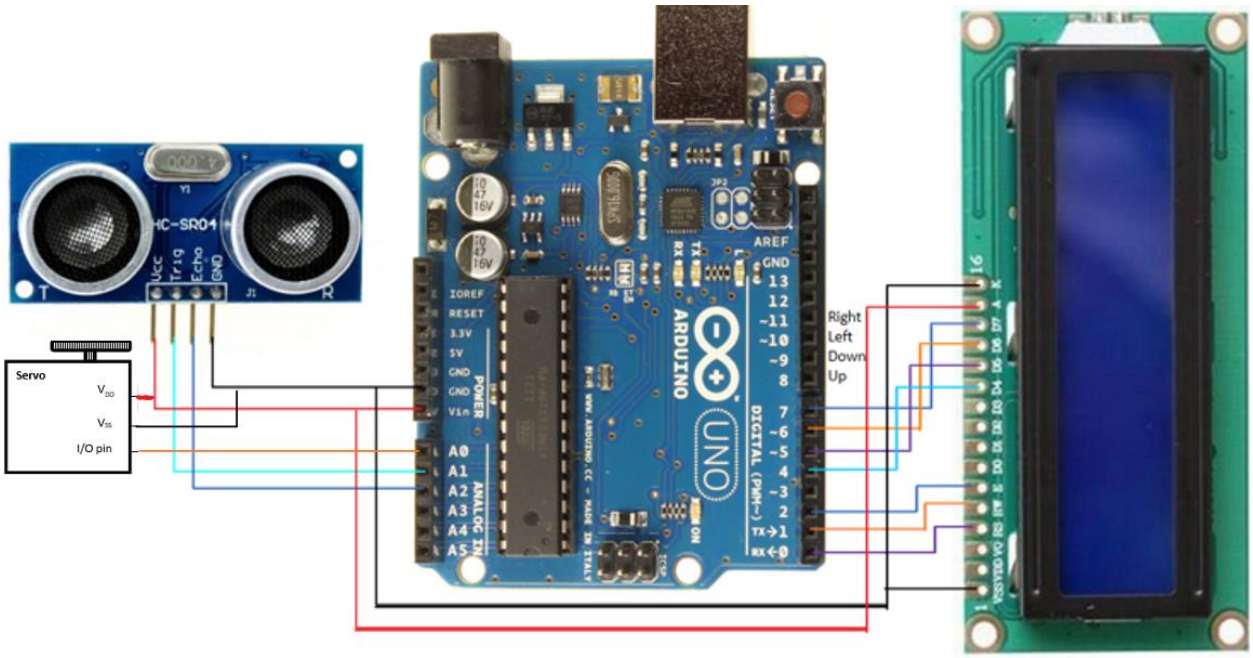

Figure 13: Phase 2 Wiring Diagram

ii. Software

In addition to the functions from Phase 1, the software design in Phase 2 included functions to handle the button inputs as well as LCD module commands. This also required changes to the main program. The new flow diagram is shown in Figure 14.

Also in addition to the software designs in Phase 1, Phase 2 included directional buttons and an LCD screen. The directional buttons up, down, left, and right were mapped to PB0, PB1, PB2, and PB3, respectively. The LCD used PORTD by mapping the Register Select (RS), Read/Write (RW), and Enable pins to PD0, PD1, and PD2, respectively and by mapping the Data (DB) pins DB7:DB4 to PD7:PD4, respectively.

The LCD required two functions: lcd_wr and lcd_cmd. Lcd_wr writes a character to the screen while lcd_cmd delivers an instruction such as clear screen, shift cursor, etc. 3 possible menus were displayed on the LCD screen including a settings menu, an information screen and a confirmation screen. Inside the settings menu are two adjustable features: Cups, which determines the number of cups that will be dispensed, and Hours, which determines the number of hours between feedings.

The buttons all create a pin change interrupt. The corresponding ISR sets a flag indicating a button was pressed and disables pin change interrupts. Inside the main program, when the conditional statements checks that the flag was set, it determines which button was pressed and which menu option is currently selected. If the Cups option is selected, up results in an increase in cups while down results in a decrease. The same is true of the Hours option. Up and down are used as an enter key if the settings menu is selected. Left and right switches between menu selections according to their direction.

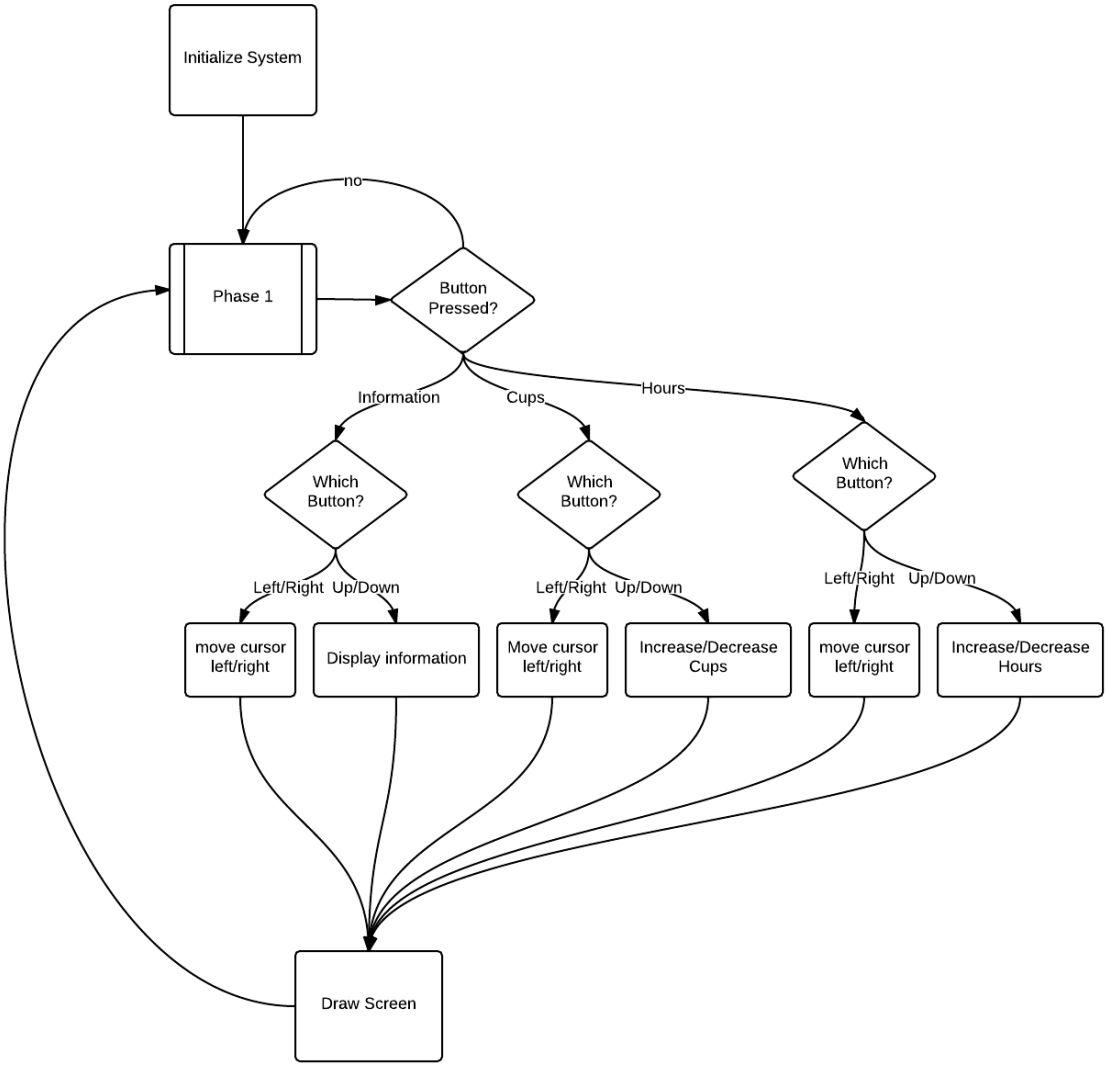

Figure 14: Phase 2 Flow Diagram

D. Phase 3

The final phase of the design cycle polished the concept of combining time and proximity to create a better pet feeder by including one final functionality: the ability to detect how much food remains in the food container. The level 0 block diagram in Figure 15 shows the added level sensor input. The level 1 block diagram in Figure 16 shows the level sensor feeding into the Atmega328P without increasing the number of outputs, where solid lines indicate feeder piping and dotted lines indicate electrical signals.

Figure 15: APFD Level 0 Block Diagram – Final Design

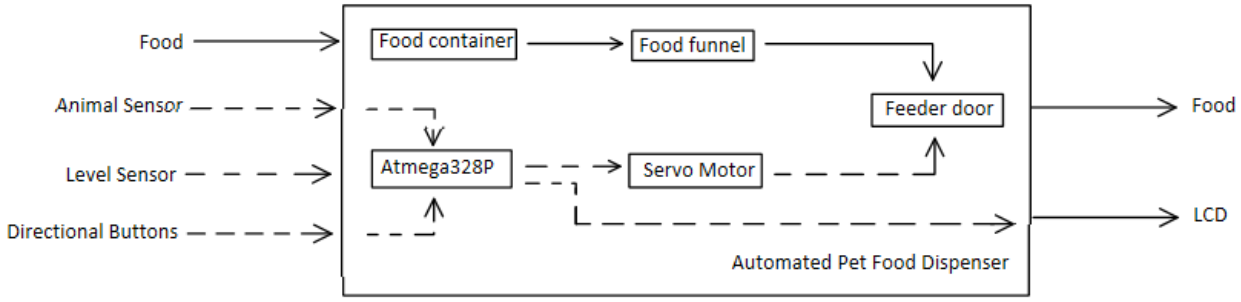

Figure 16: APFD Level 1 Block Diagram – Final Design

i. Hardware

In addition to the hardware components involved in phase 2, phase 3 included an additional 40kHz ultrasonic range finder. The new range finder is used to determine the amount of food remaining in the container. This was placed atop the container on the inside wall facing straight down, allowing it to measure the distance to the food, from which the level is determined. The updated wiring diagram for phase 3 is shown in Figure 17.

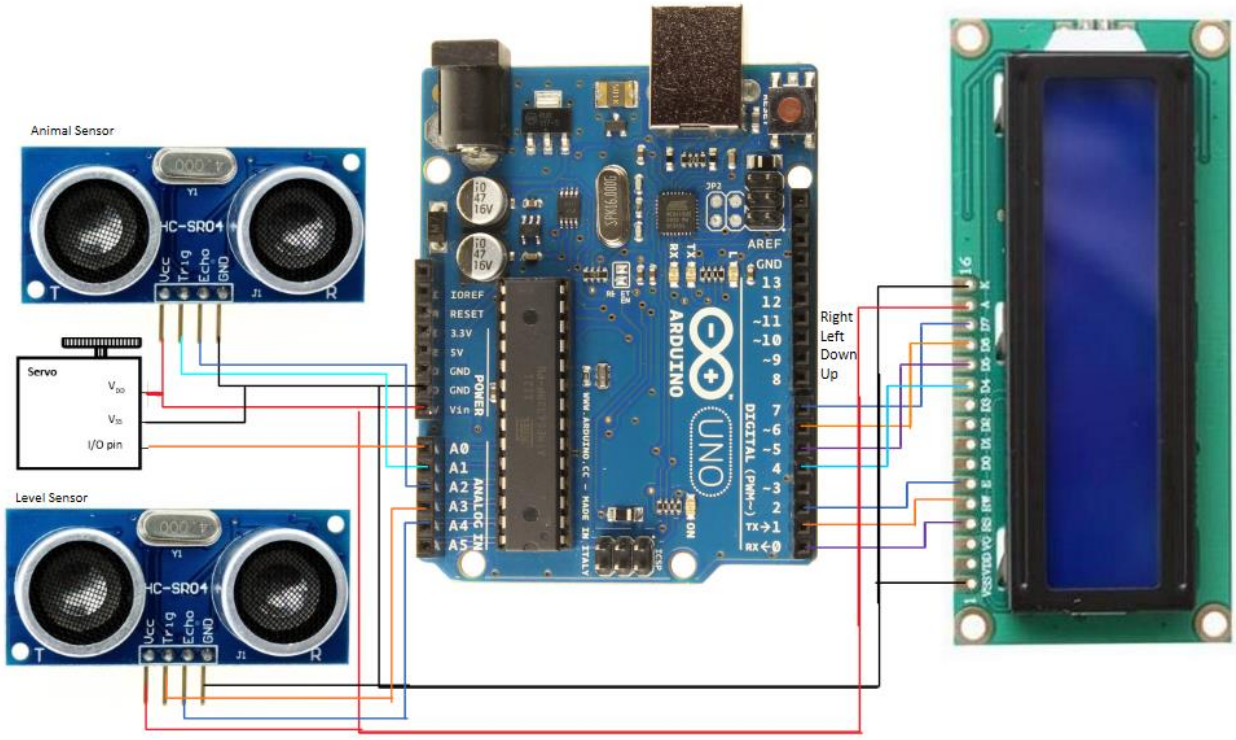

Figure 17: Phase 3 Wiring Diagram

ii. Software

The only addition made in this phase to software was the inclusion of the “Food Remaining” displayed within the Information menu option.

VI. Testing and Troubleshooting

A. Structure

To meet specifications, a cardboard box that was 1’ by 1’ by 2’ was used as the base which supported the food container that was 1’ in diameter and 2’ tall. This measured to be 1’ wide by 1’ deep by 4’ tall as required.

B. Feeding System

To meet specifications, a water bottle was selected as the container which measured 12” in diameter at the top and tapered down to 2” in diameter at the bottom. The servo operated on 5V from the supply rail and rotated 180° when it received the proper signal from the Atmega328P microcontroller.

Meeting the specifications for feeding was challenging due to the operation of the servo motor. The motor required a minimum of 15 pulses to reach the desired location. Each pulse took 20ms, resulting in 210ms to open and another 210ms to close. While this met the original specification and allows for accurate quantities of food to be dispensed, the accuracy suffers severely for quantities less than 2 cups. This trend is listed in TABLE I and depicted in Figure 18. This data was acquired by starting with a full container (60 cups) and dispensing a specified number of cups repeatedly until the container was empty and recording the amount of food dispensed each time. The data reveals important information about the device and its limitations in how it operates.

From Figure 18, it is clear that more cups in each dispensation leads to more accurate results. This is likely because there is a slight delay between the door opening and the food dropping due to static friction. Once the static friction is overcome, the rate of flow becomes more or less constant and thus produces predictable results. The effects of this as the open duration time decreases becomes evident as the 1 Cup and 2 Cups plots are observed.

Additionally, the 1 Cup plot is significantly more imprecise and inaccurate than the rest because of another flaw in the mechanical design. The 210ms required to open and close the feeder door creates a nonlinearity in the rate of flow, producing unpredictable results as the open duration time approaches half a second. Unfortunately, 1 cup of food requires 0.5s, which means that the door is in the process of moving 84% of the time, causing extremely unpredictable results as observed in the graph.

Finally, every plot experienced an increase in percent error between about 18 cups remaining to about 42 cups remaining. Observing the structure of the 5 gallon water reveals why this is the case. The diameter at the depth levels corresponding to that range is smaller than the diameter at the top and bottom. Because the cups were calibrated at a full container, this creates an error

while in the narrower portions of the container. This reveals that the rate of flow is primarily dependent on how quickly the food leaving the base can be replaced by the food resting above it. When the food level is at a narrower section, there is less food per square inch to fill in the voids.

TABLE I: Feeding Test Values

1 Cup			2 Cups			3 Cups			4 Cups			5 Cups			6 Cups			7 Cups		
Cups Dispensed	Cups Remaining	% Error	Cups Dispensed	Cups Remaining	% Error	Cups Dispensed	Cups Remaining	% Error	Cups Dispensed	Cups Remaining	% Error	Cups Dispensed	Cups Remaining	% Error	Cups Dispensed	Cups Remaining	% Error	Cups Dispensed	Cups Remaining	% Error
1.2	58.8	20	2	58	0	3	57	0	4	56	0	5	55	0	6	54	0	7	53	0
0.9	57.9	10	2	56	0	3	54	0	4	52	0	5	50	0	6	48	0	7	46	0
0.8	57.1	20	2	54	0	3	51	0	4	48	0	5	45	0	5.8	42.2	3.333333	6.9	39.1	1.428571
1.3	55.8	30	2	52	0	3	48	0	4	44	0	4.9	40.1	2	5.8	36.4	3.333333	6.8	32.3	2.857143
1.2	54.6	20	2	50	0	3	45	0	3.9	40.1	2.5	4.8	35.3	4	5.8	30.6	3.333333	6.9	25.4	1.428571
1.3	53.3	30	2	48	0	2.9	42.1	3.333333	3.9	36.2	2.5	4.8	30.5	4	5.9	24.7	1.666667	7	18.4	0
0.9	52.4	10	2	46	0	2.9	39.2	3.333333	3.8	32.4	5	4.8	25.7	4	6	18.7	0	7	11.4	0
1.1	51.3	10	2	44	0	2.8	36.4	6.666667	3.8	28.6	5	4.9	20.8	2	6	12.7	0	7	4.4	0
1.2	50.1	20	1.9	42.1	5	2.9	33.5	3.333333	3.9	24.7	2.5	5	15.8	0	6	6.7	0	4.4	0	37.14286
1	49.1	0	2	40.1	0	2.8	30.7	6.666667	3.8	20.9	5	5	10.8	0	5	1.7	16.66667			
0.8	48.3	20	1.9	38.2	5	2.9	27.8	3.333333	3.9	17	2.5	5	5.8	0						
0.8	47.5	20	1.9	36.3	5	2.9	24.9	3.333333	4	13	0	5	0.8	0						
1.1	46.4	10	1.9	34.4	5	2.9	22	3.333333	4	9	0									
1	45.4	0	1.8	32.6	10	2.9	19.1	3.333333	4	5	0									
1.3	44.1	30	1.8	30.8	10	2.9	16.2	3.333333	4	1	0									
0.9	43.2	10	1.9	28.9	5	3	13.2	0												
0.9	42.3	10	1.8	27.1	10	3	10.2	0												
1.1	41.2	10	1.9	25.2	5	3	7.2	0												
1	40.2	0	1.9	23.3	5	3	4.2	0												
0.8	39.4	20	1.8	21.5	10	3	1.2	0												
0.9	38.5	10	1.8	19.7	10															
0.8	37.7	20	1.9	17.8	5															
0.7	37	30	1.9	15.9	5															
0.7	36.3	30	2	13.9	0															
0.8	35.5	20	2	11.9	0															
0.8	34.7	20	2	9.9	0															
0.9	33.8	10	2	7.9	0															
0.7	33.1	30	2	5.9	0															
0.9	32.2	10	2	3.9	0															
0.8	31.4	20	2	1.9	0															

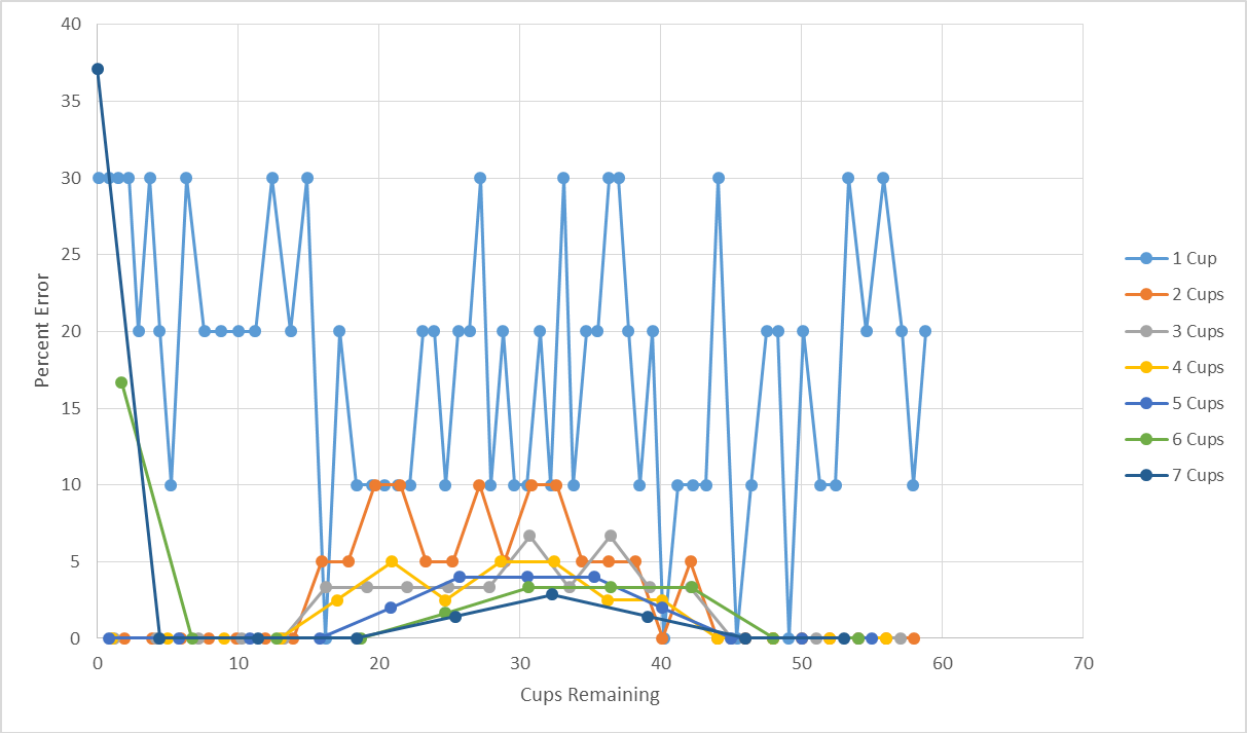

Figure 18: Feeding Test Graph

C. Detection

The detection was able to be programed to within 0.1” of accuracy at 4.5’ which prevented nuisance tripping. Nuisance tripping was possible, however, for anyone or anything passing by. This was resolved by placing the device in a corner with minimal traffic.

D. Display

The display met all required specifications. An example of the display’s main menu is shown in Figure 19.

Figure 19: APFD Display

E. Display Control

While the buttons met specifications, there was one glitch discovered. When the buttons were pressed repeatedly, the screen would update, showing that a button was recognized, but the corresponding values would not update. This occurred whenever the button was pressed more than twice a second. Despite hours of troubleshooting, no explanation has been found.

F. Microcontroller

The microcontroller has sufficient I/O ports and a proper 5V rail. The 16bit timer was calibrated to a precise second measurement. It was compared to a live clock for 24 hours and maintained its accuracy throughout. The ranging module was able to detect within a 0.1” accuracy using the 8bit timer to measure distance.

VII. Conclusion

This project successfully incorporated time and proximity to produce a superior pet food dispenser. Useful functionality was included already, though it can be modified to adapt to customer’s needs and improve its usability. The device is already almost marketable with its valuable features such as customizability of quantity and frequency of feeding, but certain improvements would have to be made before selling this product.

First, the materials used were repurposed materials and were not very durable. To properly market this product, a sleek, durable, and practical design would need to be used to manufacture the product. Also, the proximity sensors used were sufficient for the prototype, but other sensors would be more accurate and minimize nuisance tripping. For instance, an RFID tag would be used so that the device only activated when the animal and nothing else came by. This RFID tag technology would also allow for the device to service multiple pets with varying needs, even selecting from different containers with different food supplies.

Bibliography

[1] Micropik, *Ultrasonic Ranging Module HC-SR04 Datasheet* [Online]. Available: http://www.micropik.com/PDF/HCSR04.pdf

[2] Samsung, *16COM/40SEG Driver & Controller For Dot Matrix LCD* [Online]. Available: http://www.lcd-module.de/eng/pdf/zubehoer/ks0066.pdf

[3] HiTec, *Announced Specification of HS-645MG Standard Deluxe High Torque Servo* [Online]. Available: http://www.rctoys.com/pdf/hitec-servos/HIT-HS645MG.pdf

[4] Atmel, *Atmel 8-bit Microcontroller with 4/8/16/32Kbytes In-System Programmable Flash* [Online]. Available: http://www.atmel.com/Images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Complete.pdf

Appendix A: Senior Project Analysis

Summary of Functional Requirements

The automated pet food dispenser (APFD) Uses sensors to detect when an animal is present and delivers food on demand, ensuring the food is fresh and not wasted. To prevent overfeeding, a timer is used to allow food to be dispensed only once per a given time period. The quantity and frequency of feeding is user adjustable via the settings menu of the onboard LCD screen using four directional buttons.
Primary Constraints

The most limiting factors in my approach were time, money, and knowledge. There was more functionality I wanted to include in my project but I did not have the time to include it. Money also limited what supplies I could purchase and resulted in a pretty basic construction. Another factor that lead to pretty basic construction was my limited knowledge of materials and manufacturing. If I had a better knowledge base of manufacturing with access to it and time to build it, I would have had a much better product. This is was a valuable lesson to learn, as every project encounters these three constraints at some level.

Economic

Project Costs
The original estimated project costs are shown in TABLE 1. The actual project costs are shown in TABLE 2. Finally, the Bill of Materials is shown in TABLE 3.

TABLE 2: ESTIMATED PROJECT COSTS

Expenses	Cost
REQUIRED	
Motor	$40
Container	$20
Platform & structure	$20
Atmega328	$2
Arduino uno	$20
Usb charger	$5
Ultrasonic sensor	$5
Total	$112

TABLE III: ACTUAL PROJECT COSTS

Expenses (actual)	Cost
REQUIRED	
Hitec HS-645MG Servo	$31.04
5 Gallon Water Bottle	$20.33
Platform & structure	$4.93
Arduino uno w/ATmega328P	$18.10
Usb charger	$9.95
Ultrasonic sensor (x2)	$8.99
Buttons (x4)	$5.99
LCD Display	$9.99
Total	$109.32

TABLE IV: Bill of Materials

Part Name	Type	Qty	Unit Cost	Total	Manufacturer
Hitec HS-645MG Servo	Motor	1	$31.04	$31.04	Hitec
5 Gallon Water Bottle	Bottle	1	$20.33	$20.33	VMI Housewares
Cardboard Box 1' x 1' x 2'	Box	1	Free	Free	unknown
Arduino UNO w/ATmega328P	Microcontroller	1	$18.10	$18.10	SainSmart
USB Charger	Charger	1	$9.95	$9.95	PowerGen
Ultrasonic Range Finder	Sensor	2	$4.50	$9.00	SunFouder
Tactile Push Button	Switch	4	$0.50	$2.00	Microtivity
IM161 LCD Module	Display	1	$9.99	$9.99	SainSmart
Jumper Wires (M-M)	Wires	5	$0.10	$0.50	unknown
Jumper Wires (M-F)	Wires	19	$0.10	$1.90	unknown
IB830 Experiment Breadboard	Breadboard	1	$8.00	$8.00	Microtivity
2" PVC Pipe (2 feet)	Pipe	1	$3.97	$3.97	Charlotte
45 Degree 2" PVC Pipe Elbow	Pipe	1	$0.96	$0.96	Charlotte
			Total	$115.74	

Additional Costs

The only additional expenses besides those listed in the tables above included a computer from which the device was programed and a USB oscilloscope for testing and troubleshooting purposes. While I already owned both of these devices and did not need to purchase them specifically for the project, it would have added an additional $700 for the laptop computer and $100 for the oscilloscope.

Development Time

The estimated time to develop this project varied is shown in Figure 20 which varies noticeably from the actual time required shown in Figure 21.

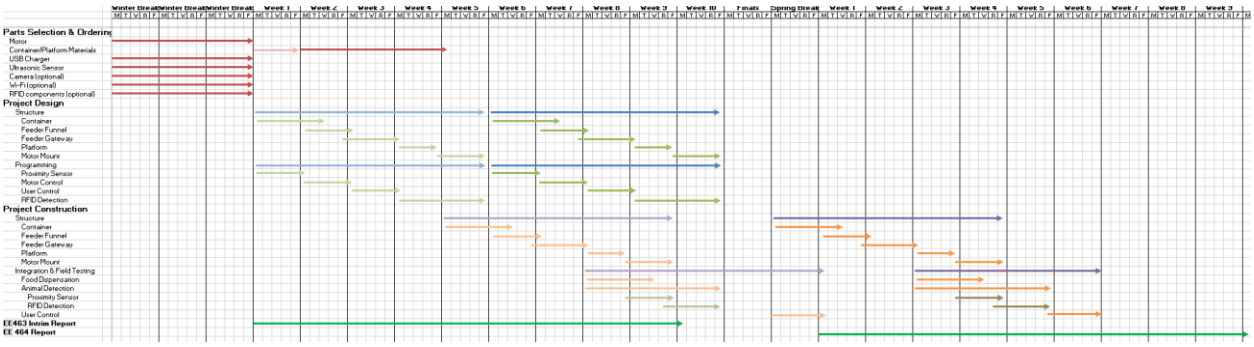

Figure 20: Estimated Development Time

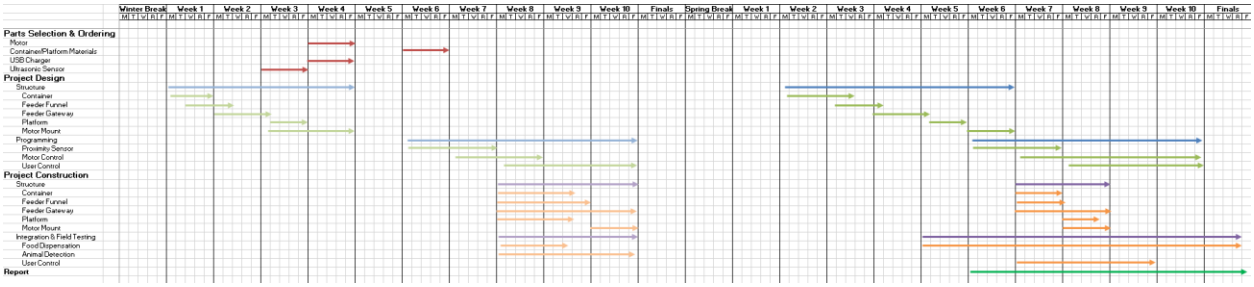

Figure 21: Actual Development Time

If manufactured on a commercial basis:

- Estimated number of devices to be sold per year: 100
- Estimated manufacturing cost for each device: $100
- Estimated purchase price for each device: $200
- Estimated profit per year: $10,000
- Estimated cost for user to operate device, per unit time: This will vary according to electricity costs. However, using worst case values, it will be less than (12mA)*(5V)*35cents/kWh = 2.1 cents per 1000 hours

Environmental

There are numerous environmental impacts this device could have. To list a few:

1. There will be a slight increase in electricity usage, both on the consumer and industrial level.
2. The extra free time people have will result in increased driving and other forms of travel.

3. The facilities to manufacture, ship, and sell the product will also affect the environment through increased pollution, traffic congestion, and energy usage.
4. Increased usage of precious metals, plastics, and other materials used in its construction, driving the mining of such materials.
5. Increased waste when the product breaks or is discarded.

Manufacturability

Manufacturing this device is still a long ways off. There is still no design for it to be officially manufactured with more than re-purposed materials. Once there is a design, however, it will still be challenging to find a plant to manufacture the device in. Also, parts would have to come from a variety of places making shipping a challenge. This all would also require a workforce or outsourcing the work.

Sustainability

This device requires little maintenance since it is plugged in and not designed to be moved frequently. The only maintenance is refilling the food container when the food runs out. This results in an efficient use of precious metals and plastics because the longevity will be much greater than most consumer products on the market today.

The prototype specifically used re-purposed materials which supports the sustainable idea that we should "reduce, reuse, and recycle." Even when manufactured properly, the materials will likely be recycled materials to promote sustainability.

Upgrades that would improve the design of the project would include the ability to serve multiple pets with varying needs. This would reduce the need for multiple devices (which would be wasteful) and would be more convenient to the user. Also, including a safety mechanism that prevents rodents or other pests from eating the food would be beneficial.

Upgrading the design would require a redesign of the whole system since each part works together. However, due to its simplicity, a redesign is not as overwhelming as it sounds. A few modifications can be made to each part affected without requiring a complete revision. Some conceivable upgrades could be simply added to an input to the microcontroller to affect only the operation, at which point other parts of the project would not need to be altered.

Ethical

The concept of the automated pet food dispenser (or any automated device for that matter) is based on the natural desire to have an easier life. This can potentially enable people to be lazy and not take responsibility. At an extreme level, the owner of the animal may be tempted to neglect his/her pet with an "out of sight, out of mind" sort of mentality. It is important that pets still receive the attention they need from their owners.

Ethical issues would also need to be considered during the manufacturing process. Work environments for the employees and salary are important factors in manufacturing a product ethically. The product should be built with integrity, resulting in a consistent product. Safety of all individuals involved is extremely important.

Another ethical consideration is in the design. This product MUST be reliable. The life of the animal depends on the device consistently providing food as expected. If any corners are cut and/or glitches in the system are overlooked, then the system may fail and result in the animal starving to death. This would result in lawsuits and other problems such as poor public branding.

Health and Safety

In addition to the health and safety concerns of neglect expressed in the previous section, this device impacts the health and safety of the consumer as well as the worker. Producing the device results in more jobs which enables increased health and safety for the worker, assuming the workplace is a safe and healthy environment. For the consumer, the device could cause potential injury through its moving parts or by it falling over. Caution should also be taken while refilling the container since pet food can be heavy and strain the persons back. Finally, if the machine malfunctions, rodents or other pests could be attracted and bring with them diseases.

Social and Political

As mentioned in the Ethical section above, this device could lead someone to start neglecting their pet. On a grander scale, if this product were to become increasingly popular, it could affect how society views and treats their pets. With more and more things becoming automated, responsibility of taking care of something is becoming more and more uncommon. Eventually, this will lead to an irresponsible society where people believe their needs should be taken care of for them.

Development

The most important development truth I learned through this project is the need to plan and act accordingly. I allowed myself to get distracted by other demands and procrastinated parts of the project. This resulted in insufficient time and supplies to make a great product. One of the more useful tools I learned regarding this was how to create and utilize a Gantt chart to assist in project planning. But the most important lesson was this: a plan is only as useful as it is used. Essentially, creating a plan may help someone to brainstorm, but if it is never actually used, the plan becomes worthless. I also gained additional familiarity with programming and testing/troubleshooting.

Appendix B: Source Code

```
/* Senior Project Source Code
 *
 * Project Name: Automated Pet Food Dispenser
 *
 * Author: Kevin Shibley and Sal Navarro
 *
 * Program Description: This program controls the Automated Pet Food Dispenser (APFD).
 *
 * Ports Used:
 * - PORTB: PB3-PB0 for right, left, down, and up, respectively
 * - PORTC: PC0 for servo signal, PC1 for ultrasonic trigger, PC2 for ultrasonic echo
 * - PORTD: PD7-PD4 for LCD DB7-DB3 (nibble mode), PD2-PD0 for LCD EN, RW, and RS,
respectively
 */


#include <avr/io.h>         // registers locations and some other things
#define F_CPU 16000000UL    // tells compiler the freq of your processor
#include <util/delay.h>     // software delay functions
#include <avr/interrupt.h>



////////////// LCD and MENU Variables //////////////
#define rs 0
#define rw 1
#define en 2

void lcd_cmd(char);
void lcd_char(char);
void lcd_wr_cmd(char);
void lcd_wr_char(char);
void lcd_init();
void initTimer0(void);
void menudraw();
void check_button();
void incmenuloc();
void decmenuloc();
void inccups();
void deccups();
void inchours();
void dechours();

       int menuloc=0;       //start menu at first position
       int menunum=0;
       int lcd_upd=1;       //initialize lcd update to 1 to enable initial draw
       char cups[] = "00 "; //note: this is equivalent to this -> char cups[] =
                            {0x30,0x30,0x20};
       char hours[] = "03";
       char mo1[] ="Cups:"; //menu 1, option 1
       char mo2[] ="Hours:";       //menu 1, option 2
       char mo3[] ="Enter:  ";     //menu 1, option 3
       char mo4[] ="Info: ";       //menu 1, option 4
       int menuop[4]={0x86,0x8F,0xC6,0xCD};
```

```
////////////// LCD and MENU Variables end //////////////


/////// Timer Variables /////
       void initTimer0();
       int counter=0;
       int topcount=997;
       int secs=0;
       int mins=0;
       int hrs=0;
/////// Timer Variables end /////


/////// Ultrasonic Variables ////////////
#define rot_time_r 100
#define rot_time_l 100
#define rot_right 1400
#define rot_left 700
int rot_delay=500;

#define servopin 0
#define trigger 1
#define echo 2
#define trigger_delay 10
#define dist_1 350
void send_trigger(void);
int send_receive(void);

/////// Ultrasonic Variables end ////////////



int main(void){
DDRD = 0xFF;  //Set all pins to out. PD7-PD4 -> DB7-DB4, PD2-PD0 - EN, RW, and RS,
              respectively
DDRB = 0x00;  //Set all pins to input. PB3-PB0 -> right, left, down, and up, respectively
DDRB |=(1<<5);       //except for PB5 (LED) for testing
PORTB = 0xFF; //Buttons are active low and LED begins on
DDRC |= (1<<servopin)|(1<<trigger);      //output to servo and trigger ultrasonic
DDRC &= ~(1<<echo);  //input for ultrasonic echo

int foodflag=1;      //this flag is set when enough time has passed

lcd_init();

TCCR1A = 0x00;       //timer for ultrasonic ping
TCCR1B = 0x03;

PCICR = 0x01; //enable pin change interrupts on any enabled PCINT[7:0]
PCMSK0 = 0x0F;       //enable PCINT3-0 (PB3-0). This allows button presses to generate
                     interrupts
initTimer0(); //initialize timer 0
sei();               //now that interrupt timers are in place, enable global interrupts
menudraw();          //begin by drawing the menu on the LCD screen


////////////////////// Program starts here \\\\\\\\\\\\\\\\
```

```
while(1)
{
_delay_ms(1);
	if (lcd_upd)  //check if button press was registered by pin change interrupt
	{
		lcd_upd=0;	//reset the flag
		check_button();	//determines which button was pressed and modifies
				appropriate variables (hours, cups, etc)
		menudraw();	//draws the settings menu with the updated values
		PCMSK0 = 0x0F;	//re-enable PCINT3-0 (PB3-0). Note: PCINT3-0 are
				disabled within the interrupt vector to prevent
				multiple button presses
	}
	_delay_ms(1);

	int tmphours= ((hours[0]-'0')*10)+(hours[1]-'0');	//convert the current Hours
								value entered by user to an
								integer
	if(hrs>=tmphours)	//compare that integer with the current elapsed time from the
				timer.
	{
		foodflag=1;	// if enough time has passed, set the foodflag to enable
				feeding
	}
	_delay_ms(1);
	if (foodflag) //if the food flag has been set
	{
		int pulse;
			pulse = send_receive();	//then check to see if something is
							present
			_delay_ms(1);

			if ((pulse < dist_1)&&(pulse>0))	//if an object is present and is
								close enough (dist1), then feed
								the animal.

	//note: the ultrasonic returns a 0 if something is out of range, hence the
	&&(pulse>0) condition.
			{
				PORTB ^= (1<<5);
				foodflag=0;		// animal detected, reset flag
				mins=0;			// reset the clock
				hrs=0;
				secs=0;
				// convert the current Cups selection to an integer
				corresponding to its open duration time.
				rot_delay = ((cups[0]-'0')*10)+(cups[1]-'0')*10;

	// as cups increase, the feeder door remains open longer, releasing more food.

	// as cups decrease, the feeder door closes quicker, releasing less food.

				for (int i=0 ; i<rot_time_l ; i++)//move servo 180
								counterclockwise
				{
					PORTC |= (1<<servopin);
					_delay_us(rot_left);
```

```
                                PORTC &= ~(1<<servopin);
                                _delay_ms(20);
                        }

                                //remain open for the user defined duration of time.
                                for (int i=0;i<rot_delay;i++)
                                {
                                        _delay_ms(1);
                                }

                        for (int i=0 ; i<rot_time_r ; i++) //move servo 180 clockwise
                        {
                                PORTC |= (1<<servopin);
                                _delay_us(rot_right);
                                PORTC &= ~(1<<servopin);
                                _delay_ms(20);
                        }
                        _delay_ms(10);      //let motor settle

                }
        }

        _delay_ms(1);
}


return 1;
}

void send_trigger()
{
        PORTC |= (1<<trigger);
        _delay_us(10);
        PORTC &= ~(1<<trigger);
}

send_receive()
{
        send_trigger();
        while(!(PINC & (1<<echo)));
        TCNT1 = 0;
        while(PINC & (1<<echo));
        return TCNT1;
}

void lcd_init()
{
        lcd_cmd(0x02);      //note: lcd reads 8bit by default. This sends 0x00 and then
                            0x20, the instruction for 4bit interface length.
        lcd_cmd(0x28);      //now that we're in 4bit, this sends two nibbles: 0x2 then
                            0x8, the instruction for 4bit, 2lines, and 5x7 dots
        lcd_cmd(0x0C);      //set display, no cursor or blinking cursor
        lcd_cmd(0x06);      //cursor moves right, do not shift display
}

void lcd_cmd(char cmd)
{
        char cmd1;
```

```
	cmd1 = cmd & 0xF0;          //Send top four bits to LCD
	lcd_wr_cmd(cmd1);

	cmd1 = ((cmd<<4) & 0xF0);   //Send bottom four bits to LCD
	lcd_wr_cmd(cmd1);
}

void lcd_char(char data)
{
	char data1;

	data1=data & 0xF0;   //send top four bits to LCD
	lcd_wr_char(data1);

	data1=((data<<4) & 0xF0);   //send bottom four bits to LCD
	lcd_wr_char(data1);

}

void lcd_wr_cmd(char cmd)   //cmd is instruction register (RS = low)
{
	PORTD=cmd|0b00001000;
	PORTD&=~(1<<rs);     //Select Instruction Register
	PORTD&=~(1<<rw);     //Select write mode
	PORTD|=(1<<en);             //Enable chip
	_delay_ms(1);        //data hold time
	PORTD&=~(1<<en);     //Disable chip
	_delay_ms(5);
}


void lcd_wr_char(char data) //character is data register (RS = high)
{
	PORTD=data|0b00001000;
	PORTD|=(1<<rs);             //Select Data Register
	PORTD&=~(1<<rw);     //Select write mode
	PORTD|=(1<<en);             //Enable chip
	_delay_ms(1);        //Data hold time
	PORTD&=~(1<<en);     //Disable chip
	_delay_ms(1);
}


void menudraw()
{

	lcd_cmd(0x01);       //clear display

	for (int i=0;mo1[i] != '\0';i++)
	{
		lcd_char(mo1[i]);    //first menu 1 option (Cups)
	}

	for (int i=0;cups[i] != '\0';i++)
	{
		lcd_char(cups[i]);   //display number of cups selected
	}
```

```
	for (int i=0;mo2[i] != '\0';i++)
	{
		lcd_char(mo2[i]);	//second menu 1 option (hours)
	}

	for (int i=0;hours[i] != '\0';i++)
	{
		lcd_char(hours[i]);  //display number of hours selected
	}

	lcd_cmd(0xC0);			//cursor set to first block of second line

	for (int i=0;mo3[i] != '\0';i++)
	{
		lcd_char(mo3[i]);	//third menu 1 option (Enter)
	}

	for (int i=0;mo4[i] != '\0';i++)
	{
		lcd_char(mo4[i]);	//fourth menu 1 option (Return)
	}

	lcd_cmd(0x0E);		// enable display and cursor

	lcd_cmd(menuop[menuloc]); // set cursor to screen locations of menu 1 options 1-4
}

void check_button()
{
	if (!(PINB&(1<<0)))
	{

		while (!(PINB&(1<<0)))		//debounce that performs action upon button
					release
		{
		}

		incmenuloc();

		return;

	}
	else
	{
		if (!(PINB&(1<<1)))
		{
			while (!(PINB&(1<<1)))		//debounce that performs action upon
					button release
			{
			}

			decmenuloc();

			return;
		}
		else
		{
```

```
                        if (!(PINB&(1<<2)))
                        {
                                while (!(PINB&(1<<2)))      //debounce that performs action
                                                            upon button release
                                {
                                }

                                if (menuloc==0)
                                {
                                        inccups();
                                }

                                if (menuloc==1)
                                {
                                        inchours();
                                }
                                return;
                        }
                        else
                        {
                                if (!(PINB&(1<<3)))
                                {
                                        while (!(PINB&(1<<3)))      //debounce that performs
                                                                    action upon button release
                                        {
                                        }

                                        if (menuloc==0)
                                        {
                                                deccups();
                                        }

                                        if (menuloc==1)
                                        {
                                                dechours();
                                        }
                                        return;
                                }
                                else
                                {
                                        return;
                                }
                        }
                }

        }


}

void incmenuloc()
{
        if (menuloc<3)      // once button is pressed, check if menuloc has reached top
        {
                menuloc++;  // increase menu location if not at top
        }
        else
        {
```

```
		menuloc = 0; // else, reset menu location to first option
	}
}
void decmenuloc()
{
	if (menuloc>0)		// once button is pressed, check if menuloc has reached bottom
	{
		menuloc--;	// Decrease menu location if not at bottom
	}
	else
	{
		menuloc = 3; // else, reset menu location to last option
	}
}

void inccups()
{
	if ((cups[1]<'9')&(cups[1]>='0'))
	{
		cups[1]++;
	}
	else
	{
		cups[1]='0';
		if ((cups[0]<'9')&(cups[0]>='0'))
		{
			cups[0]++;
		}
		else
		{
			cups[0]='0';
		}
	}
}
void deccups()
{
	if ((cups[1]<='9')&(cups[1]>'0'))
	{
		cups[1]--;
	}
	else
	{
		cups[1]='9';
		if ((cups[0]<='9')&(cups[0]>'0'))
		{
			cups[0]--;
		}
		else
		{
			cups[0]='9';
		}
	}
}

void inchours()
{
	if ((hours[1]<'9')&(hours[1]>='0'))
	{
```

```
                hours[1]++;
        }
        else
        {
                hours[1]='0';
                if ((hours[0]<'9')&(hours[0]>='0'))
                {
                        hours[0]++;
                }
                else
                {
                        hours[0]='0';
                }
        }
}
void dechours()
{
        if ((hours[1]<='9')&(hours[1]>'0'))
        {
                hours[1]--;
        }
        else
        {
                hours[1]='9';
                if ((hours[0]<='9')&(hours[0]>'0'))
                {
                        hours[0]--;
                }
                else
                {
                        hours[0]='9';
                }
        }
}


        void initTimer0(void)
        {
                TCCR0A = 0x02; // timer CTC mode
                TCCR0B = 0x03; // timer clk = system clk / 64
                TIFR0 = 0x02; // clear previous timer overflow
                TIMSK0 = 0x02; // timer overflow interrupt enabled
                OCR0A = 0xF9;  // compare timer A to 249 --> 16000000/64/249
        }

        ISR(TIMER0_COMPA_vect)
        {
                if (counter >= topcount)
                {
                        counter = 0;
                        secs++;
                }
                else
                {
                        counter++;
                }

        if (secs>=60)
```

```
	{
		secs=0;
		mins++;
	}

	if (mins>=60)
	{
		hrs++;
		mins=0;

	}

	}



ISR(INT1_vect)
{
	lcd_char('h');
	_delay_ms(1000);

}

ISR(PCINT0_vect)
{
	PCMSK0 = 0x00;		//disable PCINT3-0 (PB3-0) - will be re-enabled after button
				is serviced in main program
	lcd_upd = 1;



}
```

Appendix C: User Manual

SHIBBLES 'N' BITS USER MANUAL

By

Kevin Shibley

Senior Project

ELECTRICAL ENGINEERING DEPARTMENT

California Polytechnic State University

San Luis Obispo

2014

Getting Started

Congratulations! You just made your life easier by purchasing the Shibbles ‘n’ Bits Automated Pet Food Dispenser. While we are sure you are more than capable of setting up our product all on your own, we have included this brief user manual to help you through if you need it or just want to feel even more at ease. We hope you enjoy your new device; we know your pet(s) will.

Setting up the device

Place the device in an ideal place for feeding with minimal traffic. While this device has been optimized for home use, nuisance tripping can still occur. To ensure that only your pet activates the device, set up the feeding station away from walkways or places in your home with much traffic – a place only your pet would go. Additionally, ensure the sensor is situated towards the location where you want your animal to feed from. If the sensor faces away or towards a wall, it will not be able to be activated.

Next, fill the container with food. To do this, simply remove the lid, pour in your favorite brand of pet food, then replace the lid.

Finally, plug in the device using the provided Type B USB cable and USB charger. The power port can be found on the back of the device next to the LCD screen. You will notice the LCD screen light up and then load the main menu. In this menu, there will be four options: Cups, Hours, Enter, and Info. Use the left and right directional buttons to select one of these four options (you will see the cursor below the option that is currently selected). When Cups is selected, use the up/down directional buttons to increase/decrease the number of Cups you want the feeder to dispense per feeding. Likewise, increase or decrease the number of Hours between feeding by pressing up or down, respectively, when Hours is selected. Enter these values by selecting Enter and pressing either up or down. Until you do this, the device will remain in its previous state. Finally, view important information by selecting Information and pressing either up or down. Information displayed in this menu include: time left until next feeding, food remaining in the container, and the current selection of Cups and Hours.

First Use

The first few times you use your device, you will need to train your pet to activate the device. When your pet comes to you begging for food, simply lead it near the device so that he/she trips the sensor and activates it. Repeat this until the pet no longer bugs you to feed it. That's it! Simple, right?

Maintenance

Whew, that set up was challenging, right? Yeah, we didn't think so either. Since we figured you're buying our product to make your life simpler, we made sure to leave you with easy maintenance too.

Occasionally, the food container will run out. Unfortunate, we know. Don't fret though, this can be easily remedied by following these three simple steps: 1) remove the container lid, 2) fill the container with your favorite pet food, and 3) replace the container lid.

Also, it may be sanitary to clean the device once in a while. This is really up to you, but we have noticed that crumbs can accumulate over time and attract pests in place of your pets. To clean the device, simply unplug the device and rinse out with soapy water; scrub as necessary. Finish the cleaning with a final rinse of clean water and allow the device to dry before plugging it in again.

That's it! Enjoy your easy living.