Feed Beep
(Automated Pet Feeder)

Inventors:
Alvaro Naranjo
Slava Likhachov

Introduction

When we first began brainstorming possible project ideas, we didn't want to create a simple contraption that we would end up throwing away at the end of the course, but something that would have real-world applications. We had quite a few ideas, such as a robot that would walk across windows and clean them or a car that would follow different coloured lines on a track, but we quickly realized that these would either be too complicated or would cost more than what we were willing to spend. Thus, by a process of elimination, we ended up with our current project: the pet feeder. We thought that many people who own pets aren't always around to give them food, so an automated feeder that would dispense food at user-specified intervals or times would be the perfect solution.

Results

Hardware

Initially, we were going to just use a box with a hole in it to dispense food, but we realized that it would have to have some sort of funnel so that food wouldn't become trapped in the box. It would also need some support system to keep it in place, and a latch to keep the food from spilling out at the wrong time. We also wanted it to be a prototype of a commercial product, so it would have to be viable in a practical sense as well.

The initial dimensions of the container were to be 20x20x10cm, with a funnel that was 10cm high. We wanted as much food to be stored as possible, so that the owner wouldn't have to refill the container often. After all, what was the point in an automated feeder if you had to constantly refill it? However, once we began gathering the materials, we realized a few problems. First, although we wanted to make it out of wood, that would have been too expensive. Second, the box would be too large without an even larger support system, and neither of us wanted to carry an enormous box around. If this were to be a commercial product, space would be an issue, so we made the container smaller. The container ended up having one quarter of the originally intended volume, and the funnel had a much gentler slope.

We had a few ideas for what the support system would look like. We were contemplating having it mounted on a wall, but that would be difficult to implement and would have the added problem of food potentially spilling onto the walls. We also considered having the container itself rotate instead of having a latch, but that would've required two large and expensive motors, and so the idea was abandoned. Eventually, we settled on having it mounted on legs, sort of like a chair. Initially, there would be four legs, one on each side so that the container would be fully supported. However, that would make it nigh impossible for the pet to access the food. After some experimentation and a lot of Duck tape, we decided on having four legs, with two on either side of the container. This wasn't very stable, as the legs were fairly thin and contacted the ground on an angle, so we added extra "padding" onto them in the form of Play-Dough to keep the container from wobbling.

There were a lot of ways to implement the latch. We could have it be on the inside of the container, and have it pull away so that food would drop. We could also have two separate latches on the outside that would open like doors. In the end, we decided to have a single latch on the outside that would swing open and closed. Initially, we were going to have it rotate 90 degrees so that the food would easily fall from the container, but it kept hitting the supports, so we changed the amount it rotated to 45 degrees. Finding the appropriate strength of the motor and the right type of motor took some time as well.

Software

Aside from the physical container for the food, we also needed the system that would control it. We were going to allow the user to select the time or time interval and the size of the bowl that the food would be going into. We were also planning to allow users to save their settings on a USB, but that proved to be too complicated. Since we didn`t have a clock that would run on an external battery even when the pet feeder wasn`t on, we assumed that it was always on, and that every time it was turned on it would start from midnight. We were supposed to have the machine beep when it was time for the food to come out, but that stopped working when we copied the Verilog files onto the USB to bring to the lab.

The FSM we ended up with was a simplified version of the original one, and is best described in the appendix.

Methods

Hardware

We started creating feed beep by identifying the hardware we needed. In the first two weeks we bought all the parts. Most importantly we learned to control a DC motor using an h-bridge. Then we learned to control the motor with the DE2 board using GPIO pins.

To create the box, we took a vaguely rectangular one that was already lying around in my house. This way, we wouldn't have to reinforce the structural integrity of the container itself, as the manufacturers of the box had already done that. There was already a lid on the top to put food in, but the bottom was sealed shut. Initially, we were going to just cut out the bottom and attach a funnel in its place, but then we had a better idea. We carefully unglued the flaps at the bottom, and marked a rectangle 2cm wide in the middle of each flap. Then we drew a line through the middle of the rectangle, perpendicular to the previously drawn lines. We then drew another line from the intersection of the line through the middle and the line going along the length of the rectangle to the corners of the flap. Then, we cut off the bottom half and the outside triangles of the flap, thus creating the sides of the funnel. Unfortunately, only two of the flaps were large enough to be used as sides of the funnel, so we had to cut out the other two flaps from another box. Using trigonometry and a ruler, we drew out the remaining two flaps. We left some extra space on the sides of these two flaps so that their outer edges could be glued to the inside of the flaps already attached to the box. We then glued everything together and wrapped it in enough Duck tape to hold back a charging bull. We also placed tape along the inside of the container, so that the funnel would be structurally supported from all sides. Some more tape was added to keep the ends of the tape attached to the box, as the pieces that wrapped around corners kept detaching themselves. Finally, I painted the front and top of the box, as that was the only things I could paint with the paint I had on hand.

To create the supports, we took a bundle of thirteen barbecue sticks and wrapped them in Duck tape. We then attached the ends of two of them into a V-shape, so that the supports would be more stable and would support each other. That part took a while, as the angle between them became too narrow as more Duck tape was applied. We alternated between one strip that would go around the side of the V and one strip that would go over the point. After about six layers of tape altogether, the supports were structurally sound enough that they didn't come apart. We then taped the supports to the middle of the thinner side of the box, as the supports would have more of an effect there than on the wider side. The lid opened on the wider side as well, so the supports would have gotten in the way. Despite our best efforts, the supports weren't all at the same height, and so the container was still wobbly. To rectify this, we added balls of Play-Dough to the ends. This had the added benefit of making the structure less liable to slide, as Play-Dough has a fairly high coefficient of friction.

To create the latch, we used a servo motor with a sheet of metal attached to it. We didn`t need a particularly strong or fast motor, as the sheet of metal wasn`t very heavy. We drilled a hole in the corner of the metal, then used a screw with a bolt to secure it in place. We were contemplating using a strong glue instead, but we already had the screw and other materials. We attached the motor to one of the supports using plastic straps. Then, we had to figure out how long we had to enable the motor for it to open 45 degrees. Using the rpm of the motor and the clock speed of the board, we calculated how many clock cycles we would need for the latch to open. However, it turned out that theoretical calculations are very different from practical applications, and so after much experimentation we arrived at the necessary amount. Since the motor would be going against gravity when it would close the latch, we added more clock cycles to that part of the operation. We tried adding an extra fraction of the clock cycles needed to open the latch, but Verilog doesn't support decimals, so we had to add a constant to get the latch to close instead.

Software

We continued creating our machine by thinking and designing the finite state machine that would control the device. We wanted a practical device so we modelled it intuitively. In the initial FSM, feed beep was able to set different meals in the day, (Breakfast, Lunch and Dinner), and you were able to set a timed interval if the preset feeding intervals didn't meet your needs. We soon realized that coordinating subcircuits for different meals would be too hard so we simplified it to handle just one meal.

We then tackled the creation of the subcircuits and went back to modify the FSM depending on the new problems we encountered. For that we had to model three aspects of time: setting it, running it and displaying it. We modelled time as a 24 hour with two digits for hours, two for minutes and two for seconds. Each digit would code to a hex that we would then display. We first solved the time display problem by creating display_time, a circuit which takes in all the time digits in a 24 hour clock and outputs them to seven segment displays 7 to 0, with HEX[1] and HEX[0] blank.

We then solved the problem of setting time. Initially we wanted to set digits for hours and minutes directly using switches, but we soon found this would require too many switches and possibly entering in binary or hex format. This is not intuitive for the user so we changed the design to set time using a button that would increment the time by one with every press. Now if the user wanted to set an hour or a minute they just had to push a button to add a unit of time. Of course we made sure hours and minutes don't overflow past 23 and 59. To do this we created the circuit set_time.

Now we knew which time the user wanted a clock to be started on, but we still needed to run the clock using this time. We then created run_clock, which is essentially a digital clock that will start running based on a given initial time, once this given initial time is loaded. Now we had to make the machine open the latch at the given meal time. For this we created the executer circuit. When the current time is equal to the meal time, the circuit will send signals for the motor to open the latch, delay it so that the food would have time to spill out, and then close it. The delay can be set by the user using SW[3] in the state of the FSM that controls meal size. To feed the pet, Feed Beep would do the same thing at the same time every day. To implement executer we had to think of a counter on top of an "operating clock" (CLOCK_50). Motor enable, and motor terminals were set according to the value of the counter. The intervals for the counter were calculated from the revolutions per minute (RPM) of the 3v DC motor to get us the angles of elevation and depression to open the latch. In this way we were able to control exactly how much we wanted the motor to rotate and exactly how long we wanted to delay it dispensing food before closing the food gate.

Finally ,we incorporated everything into the file control.v that sets up Feed Beep's FSM, set_time, run_clock and executer sub circuits. We also used external code: the code for an LCD display (taken from John Loomis) to display a message about the FSM's current state and the code for speaker control (taken from Altera) to beep to call the pet for a meal.

Discussion

What We Learned

In the end, everything we implemented worked perfectly. We didn`t manage to implement everything we wanted or how we wanted it, but we did manage to create a fully functioning pet feeder.

The part that took us the longest to figure out was how to get the motor to work. We thought it would be a simple affair of turning it on or off to get it to run, but it ended up being more complicated than that. It turns out there are three different kinds of motors: servo, stepper and another whose name I can`t recall. A servo motor turned out to be the simplest and most suited to our needs, since we didn`t need the precision of a stepper motor. That, however, was only the beginning. We asked the store manager for assistance with how we could get the motor to run, but he was less than helpful. He did, however, explain that we needed a PWM signal to tell the motor what to do. We didn`t know what a PWM signal was, but it turns out that we had already done something similar in the labs, where we just modify the length of a clock pulse. We still didn`t know how to control the motor, however. A TA ended up advising us to use an H-bridge. After a search on Google and some experimentation, we figured out how the H-bridge worked. Depending on whether you send low or high signals to the H-bridge, it would turn the motor in a different direction. Low in one input and high in the other would turn it clockwise, while the opposite input would turn it counterclockwise, as the inputs would connect different parts of the bridge. Giving the same inputs (high and high or low and low) would do nothing, as it would connect either both the top or both the bottom parts of the bridge, not allowing the motor to turn at all.

We also learned that no product is ever as simple as it seems. A pet feeder is not a complicated device - it`s just a motor on a box with a finite state machine controlling it. Even something that apparently simple took a lot of time and effort to make work, however, so we can only imagine how difficult making something more complex would be.

We also learned that building even something as simple as a pet feeder requires entirely too much planning. You have to consider the overall design of the machine, then change it to fit time and money constraints, then modify it some more as you come up with ideas along the way, then modify it again as you come up against obstacles you hadn`t already considered. It`s an ongoing and painstaking process, but necessary to create a functional product. The more that you plan before you start building, the fewer modifications you will have to make as you begin building the actual device. Even with the amount of planning that we did, we still did not have enough time to implement everything the way we would have liked.

Future Improvements

Next time, we would probably start building the project sooner, as it took more time than we had originally anticipated. It would also have been nice to make a container out of a material that is more sturdy than mere cardboard (like wood), as well as supports that were thicker and better designed. We would use a better adhesive than Duck tape to hold the entire structure together, as the left side began to detach after a few days and had to be held to the body of the structure with an ingenious solution of a string of rubber bands wrapped around a ballpoint pen.

We had to simplify the FSM. An extension of feed beep would allow setting different meal times, for example breakfast, lunch, dinner. Or if your pet eats a lot, setting a timed meal that will dispense food based on a timer not based on a global clock.

It would’ve been cool to save meal settings (times and sizes) in a USB drive so that you could take your USB with you and plug in this setting into any other Feed Beep machine. That way your settings are more portable and this would improve the commerciality of Feed Beep. Pet hotels for example would find it very useful to have a Feed Beep machine in a pet’s room that would feed the pet just how the owner wants.

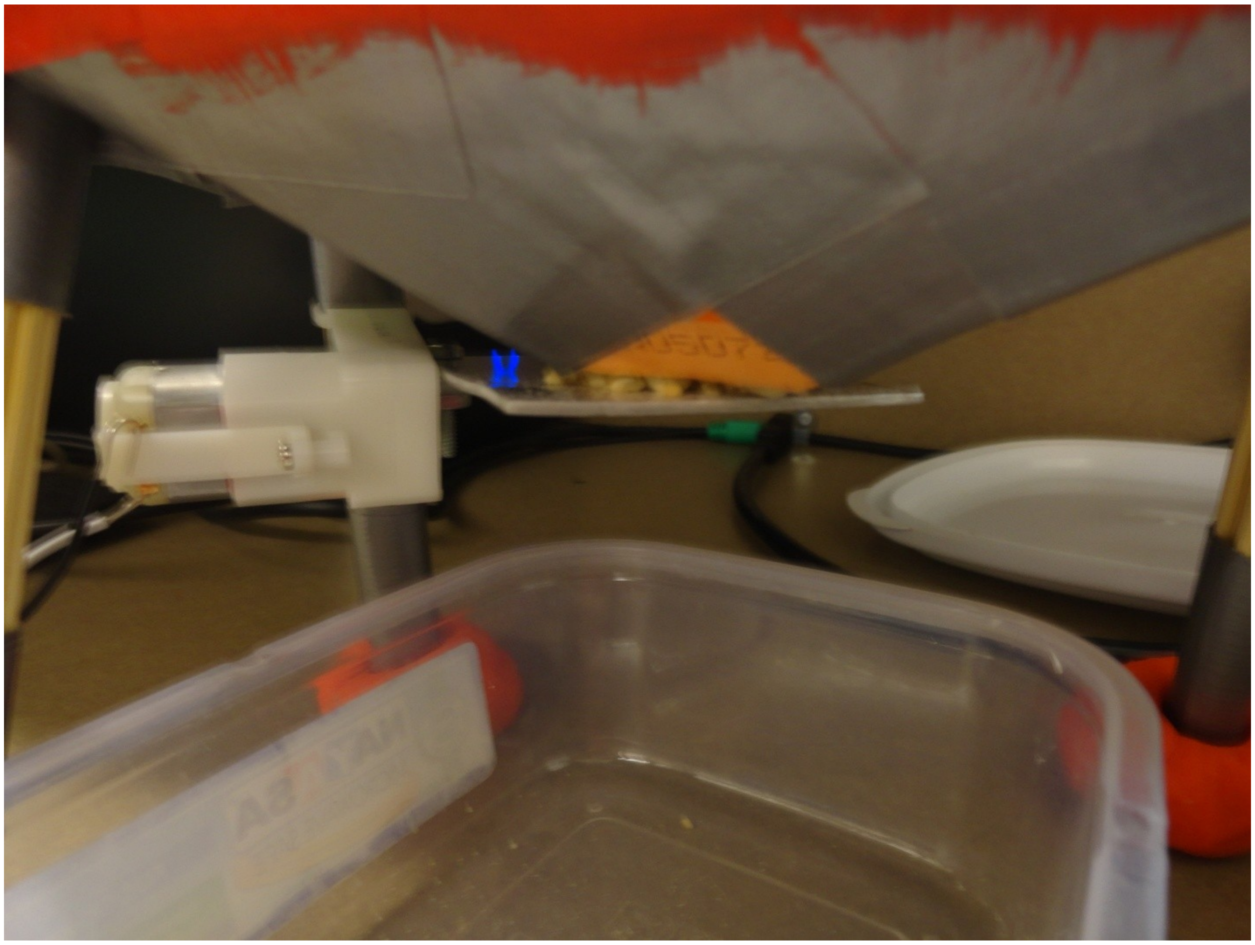
Why this project is so cool that somebody outside the course would think it is interesting

Hooray for obnoxiously long subsection titles. Since we didn't say this explicitly, here's a small section explaining it. As was stated earlier, we wanted a product that would be commercially viable, something that could be used outside of just CSC258. Lots of people have pets (for inexplicable reasons), and so this project would be something that would potentially interest lots of peoples. We'd have to make it look pretty and more professional, but we essentially created that prototype of a marketable product. I think that's pretty neat.

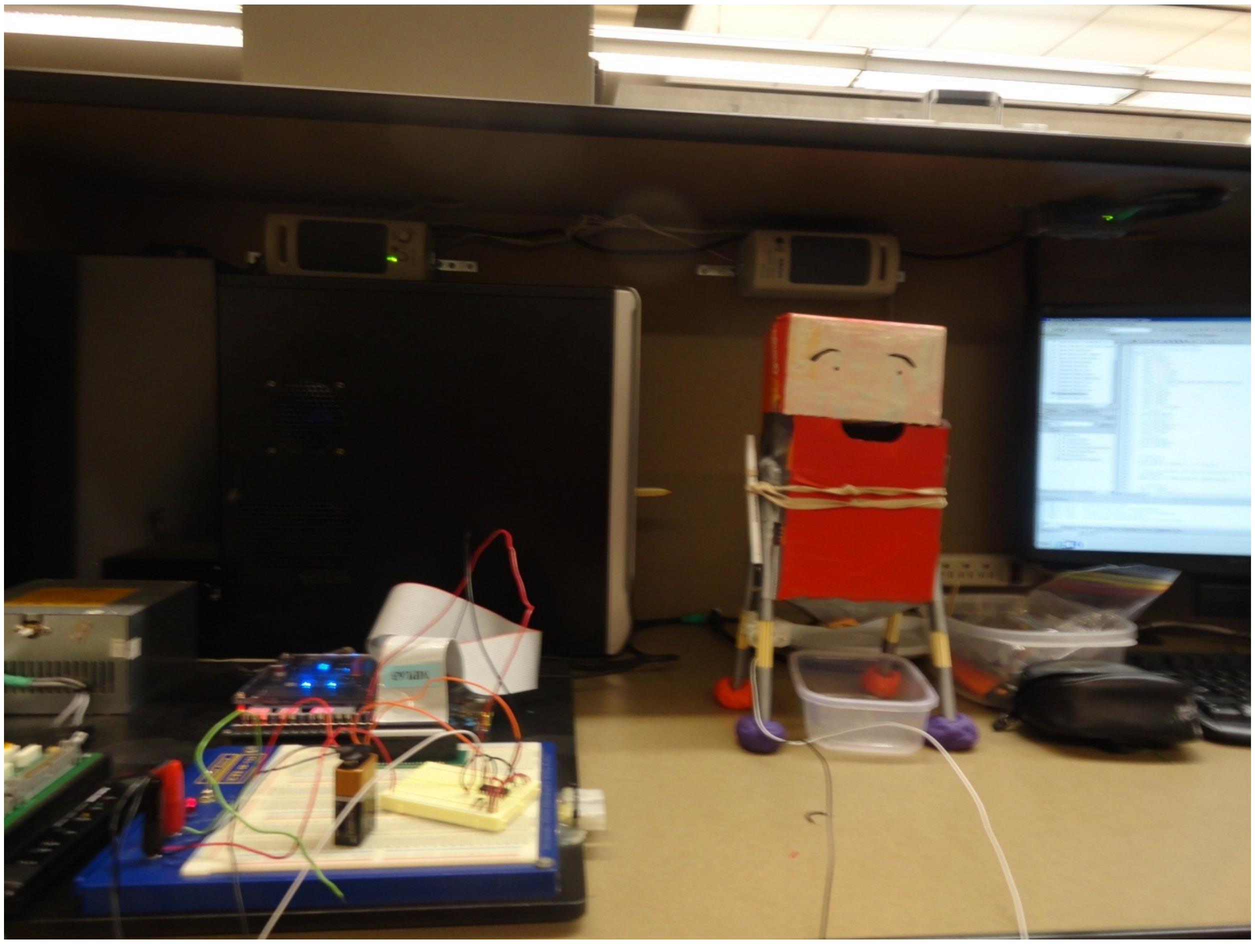
The way this builds on the course is interspersed throughout. The FSM, the motor control, the clock, the mountain of Verilog in the appendix, all attest to the fact that we weren't asleep during the course.

Appendix

Images

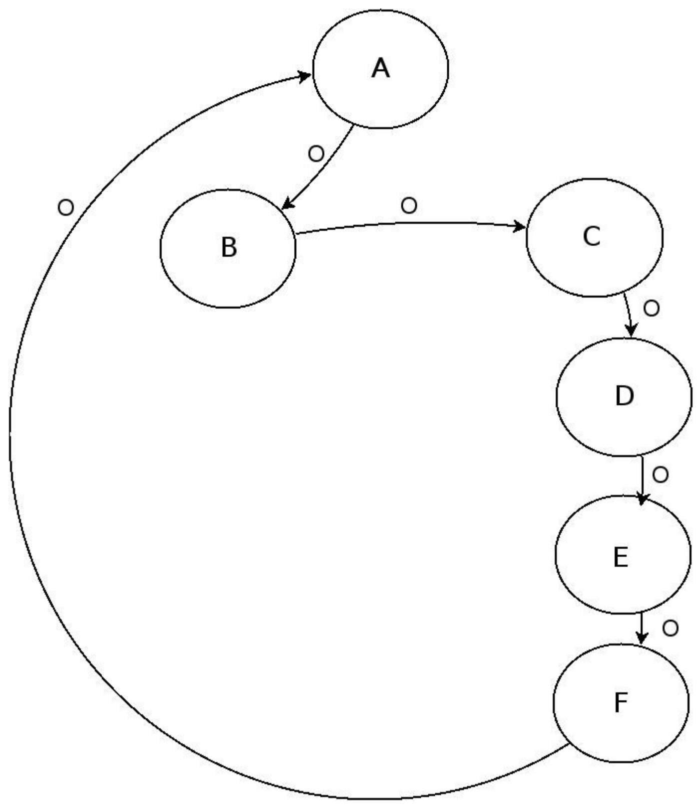


The motor, in surprisingly awful image quality. Must be the shaking of our hands as we nervously awaited judgement. Yep, that must be it. Nothing to do with our abilities as photographers.



The entire product, connected with strings of multi-coloured spaghetti to the breadboard. Much better quality than the previous image, but still awful.

Finite State Machine


$$\begin{tikzcd}
A \arrow[d, "O"] & & \\
B \arrow[rr, "O"] & & C \arrow[d, "O"] \\
& & D \arrow[d, "O"] \\
& & E \arrow[d, "O"] \\
& & F \arrow[uuuull, bend left, "O"]
\end{tikzcd}$$

State Name	Letter Code	State Code
Start	A	00000
Set Clock	B	00001
Set Meal	C	00010
Set Time	D	00100
Set Size	E	01000
Done (call executer)	F	10000

State Table (we are not showing all other transitions in the diagram).

State	Transition	Next State
A	O	B
A	ALL OTHERS	A
B	O	C
B	ALL OTHERS	B
C	O	D
C	ALL OTHERS	C
D	O	E
D	ALL OTHERS	D
E	O	F
E	ALL OTHERS	E
F	O	A
F	ALL OTHERS	F

Transitions and Operations

Name	Letter Code	Hardware	Description
Ok	O	KEY[0]	Manual Clock to transition the FSM
Add Time	Add	KEY[1]	Adds Hours of Minutes depending on S_H, S_M
Display Set/Current Time	SDisp	SW[0]	If 0 it shows time being set. If 1 it shows current time.
Set Hour	S_H	SW[2]	If State==B set S_H high and S_M low to set hour
Set Minute	S_M	SW[1]	If State==B set S_H low and S_M high to set minute
Set Size	S_S	SW[3]	If low, get standard size, small. If high, get large size. This controls how much time the food gate is open.
Run Clock	RClk	KEY[2]	Once an initial time has been set, press RClk to run a clock on starting that initial time.
Run Executer	RExe	KEY[3]	Once a meal time and a size have been choose, press RExe to run an executer, that will open food for the pet at meal time, every day.

INSTRUCTIONS OF OPERATION:

1. Start Feed Beep (load programmer). State = A. Press Ok.
2. State = B, now you can update feed beep with your current time. Have S_Disp off so you can see the time you are setting. To set minutes have S_M on and S_H off. To set hours have S_M off and S_H on. Once initial time has been chosen, press R_Clk to start running the clock on that time. Turn S_Disp on to see current time. Press Ok.
3. State=C. Only message displayed. Press Ok.
4. State =D. Now you can set a meal time. Similar to instruction 2. But don’t press R_Clk. Press Ok.
5. State=E. Now you can set a size. Turn S_S on for large size, otherwise standard size is small. Press Ok.
6. State=F. Now you are done so press RExe to run an executer that will control the food gate and open at the set time for set size and feed your pet each day. You can leave or you can press Ok to run go to Start and start again.