Frapparduino

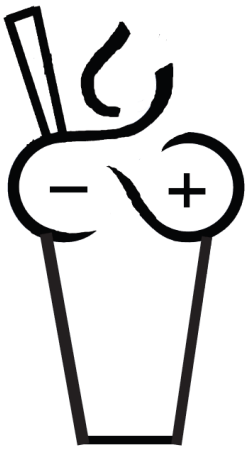
School of Engineering and Applied Sciences
Harvard University
33 Oxford Street
Cambridge, MA.

Description: A semi-automatic frappuccino maker that dispenses the appropriate amount of ingredients into a blender which automatically powers on and off after the user both selects a frappuccino flavor from a web interface and manually adds ice.

By: Student A, Student B, Student C

Class: ES50
Instructors: Marko Loncar & Christopher Lombardo

December 2014



ABSTRACT

Frapparduino is a semi-autonomous frappuccino-making machine. We use an Arduino Uno to control a series of solenoids attached to inverted bottles which are filled with the ingredients necessary to make a frappuccino: milk, coffee, caramel syrup, etc. A JavaScript program directs the Arduino to open the solenoids for different amounts of time, depending on which kind of frappuccino (Mocha, Caramel, Coffee) the user selects using our web interface. We use a wireless remotely operated switch to power the blender for the appropriate amount of time so that the mixture is blended to perfection.

INTRODUCTION

The overall goal of this project is to increase the efficiency with which a frappuccino is made.

We chose this project for a variety of reasons - the first being that we all really enjoy drinking frappuccinos, and we thought it would be interesting to create a machine that could automatically make the drink. We also harbored the secret hope that we might be able to keep our machine after ES50 so that we could make frappuccinos in our rooms at the click of a button...

In addition, we have noticed that, at ES50 project fairs, projects involving food have been incredibly popular. The TiniBot ES50 final project featured at the Spring 2014 ES50 Project Fair actually provided the inspiration for our project, and we were excited to improve the machine by interfacing a blender with the bottle-dispensing functions so that we could create more complicated drinks.

Finally, and most practically, we chose this project because we feel that the time that could be saved by Starbucks and other coffee shop baristas with the implementation of a semi-autonomous frappuccino machine, like the one that we created for our project, is most certainly non-trivial. Student C works as a barista at the Lamont Library Cafe and knows first-hand about the disproportionate amount of time required to make a frappuccino for customers relative to the amount of time necessary to make a steamed espresso drink or to pour a cup of hot coffee. Frappuccinos require several ingredients to be mixed in exact proportions, and the locating and mixing of these ingredients takes time. A machine like Frapparduino has the potential to greatly reduce the wait time for customers in line at coffee shops who have either ordered a frappuccino themselves or who are waiting behind someone who has.

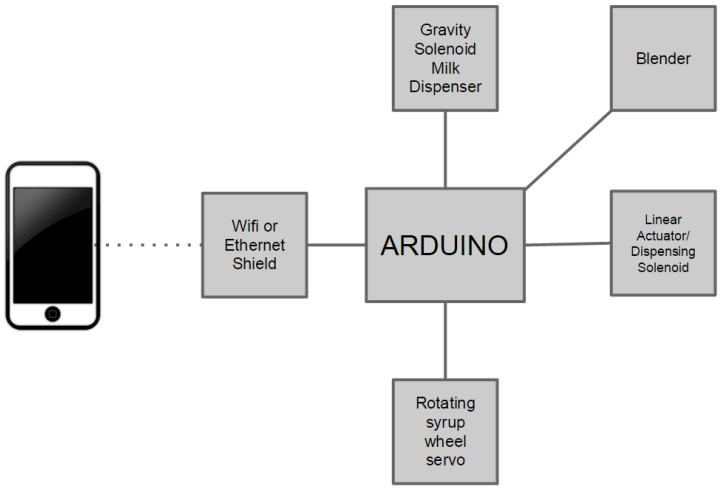
Frapparduino is an interactive project. By allowing the user to select their recipe from a web interface, to see their drink made in front of their eyes, and to consume the beverage after it's done, we tried to make this project something that people could engage with on many levels. As far as the inner workings of the machine: both the solenoids that dispense ingredients from the inverted bottles and the remote control that powers the blender are controlled by relays that are switched on by an Arduino Uno.

DESIGN

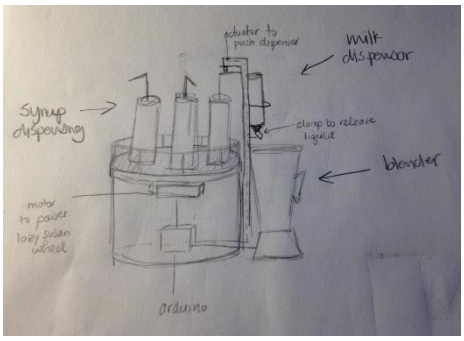
This section will discuss four aspects of design that were integral to the construction of our machine.

(1)The first design decision that we had to make had to do with the placement of ingredients with respect to the blender and the dispensing of these ingredients. We at first considered a Lazy-Susan-like wheel that would support and turn the bottles of ingredients as proper proportions of ingredients were pumped into the blender.

Here is our original flow diagram:



And the following drawing shows our original plan for the machine:

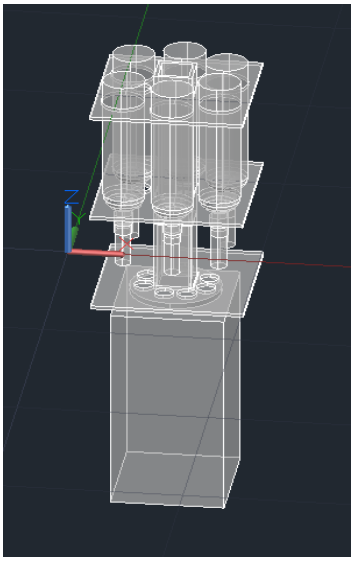


As the drawing shows, the syrups would be housed in bottles with push-dispensers on the top. At the command of the Arduino, a linear actuator would push the nozzle down until the correct amount of liquid had been squirted into the waiting blender. Then, the Arduino would tell a Servo to rotate the wheel on which the syrup bottles were mounted so that the next bottle could be moved to underneath the actuator. To dispense the milk, we envisioned mounting an inverted bottle over the mouth of the blender and using a gravity-feed solenoid to allow the right amount of milk to flow through. We did not believe that we could use gravity-feed solenoids to dispense the syrups because the syrups are very thick, and the type of solenoids we were planning to use were not designed to handle liquids of high-viscosity.

We abandoned this original design for two main reasons. First, we struggled to find linear actuators capable of pushing the pumps on the bottles with the required power and depth. Many of the linear actuators within our budget moved slowly and only reached a maximum of several inches. Stronger actuators with larger reaches had prices that climbed to the hundreds, so they were simply not feasible. Second, we decided that the Lazy-Susan architecture would simply require too much mechanical engineering to implement, especially considering the weight of all the ingredients that would rest on top of the wheel.

Therefore, taking notes from the TiniBot project of Spring 2014, we decided to only use gravity feed solenoids to dispense the ingredients. To manage the syrups which, in their raw form, might be too viscous for the solenoids, we decided to dilute the syrups with water until they flowed properly. Our design for the structure then changed to what is indicated in the following drawing:

The cleaner AutoCAD model is below:



So, the ultimate design with which we settled features inverted bottles which are open at the top, that are then connected to solenoids which dispense the ingredients in the bottles into tubing that flows, with the guidance of a tiered acrylic structure, into the blender. A chute in the middle of the structure both supports the acrylic tiers and acts as an ice chute through which the user manually inputs ice. The blender rests underneath the bottles.

(2)The second design decision with which we were faced had to do with the precision of the volume of ingredients that our solenoids would dispense. We at first planned to use two gravity feed solenoids per bottle. The first solenoid would be attached to the head of the bottle, tubing would then be fitted to this solenoid, and then another solenoid would

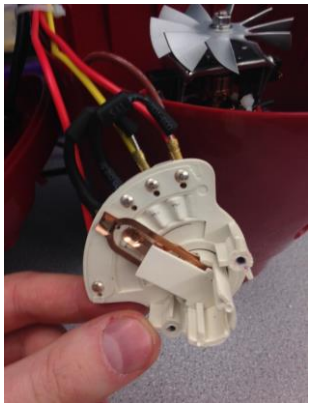
connect this tubing to more tubing that would feed the ingredient into the blender. The idea was that the tubing connecting the two solenoids would be of the appropriate length for each ingredient so that the top solenoid (the one immediately connected to the bottle) would first open and allow ingredient to flow into the tubing, the top solenoid would close, and then the bottom solenoid would dispense only the ingredient that was caught in this tubing so that an exact, measure amount of ingredient would be dispensed.

Unfortunately, when we tested this double-solenoid method, we realized that nothing would flow through the bottom solenoid if the top solenoid was closed because the tubing was completely sealed, and therefore airtight; the ingredient remained trapped in the tubing when the bottom solenoid was opened.

Thus, we abandoned the double-solenoid method and settled with the slightly less precise dispensing method of dispensing ingredient through one solenoid for a set amount of time. The following photo shows how the bottle was connected to the solenoid:

(3)A third design decision that we had to consider was how to automatically power the blender on and off.

Our original idea was to exploit the existing power dial that came with our blender. Student B opened up the blender (and shocked himself with 120V) to discover the inner workings of the blender that corresponded to the dial on the side of the blender which has the following different modes: Off, Dispense, Smoothie, Icy Drink, etc. We hoped to use a Servo to turn the dial to different modes, depending on what the drink required. The following photo shows the deconstructed dial:



When the user turns the dial, the metal tab makes contact with one of the five metal circles, completing the circuit that corresponds to a specific “mode”. We attached a Servo to this dial using poster gum and tape, as shown in next image:

Although the Servo was strong enough to turn the dial to different modes, it only did so when the Servo box and the dial were securely mounted to immovable objects. If either of the two were not mounted, then the metal tab would stay in place while the Servo box rotated. While we were struggling to figure out a way to securely mount both parts on our physical structure, Marko gave us the idea to keep the blender permanently on and to use a wireless remote on a special outlet to control the power that was applied to the blender. The next photo shows the remote control that we used and the accompanying power outlet:

Click to open expanded view

Thus, we opened up a remote control and connected the “on” and “off” buttons of this remote control to two separate relays that received signals from our Arduino Uno. With this, the Arduino Uno was able to electronically power the blender on and off.

(4)A final design decision that we had to make was the actual interface of how the user would request a drink, and how this input would be translated into instructions for the machine.

Since we were using an Arduino Uno (which does not come standard with Wifi - and we were already over budget), we decided to communicate with the Arduino through USB cable.

The code is available on Github, and basic guidelines of operation can be found here.

To summarize how the system works:

1) A web-server has been set up using the Express framework.
2) The website is served to localhost port 3000
3) The Arduino is running Standard Firmata.
4) Our web server uses the Johnny-Five Node.js library to communicate and control the Arduino via USB.
5) When the user selects their preferred drink, a window pops up asking them to please insert 1 cup of ice and then confirm to proceed.
6) This webpage communicates with the server using socket.io
7) The appropriate relays are then triggered so that the correct syrups are dispensed for an appropriate amount of time.
8) After which the blender is turned on to blend for the correct amount of time, and then turned off.
9)

The following is a screenshot of our online ordering form:

PARTS LIST

**Parts found in the lab are indicated with an asterix.*

Part	Vendor	#	Quantity	Unit Cost	Total Cost	Link
1/2" Gravity Feed Electric Solenoid Valve	Ebay	DDT-CD-12VDC	5	$13.11	$65.55	http://www.ebay.com/itm/1-2-Gravity-Feed-Electric-Solenoid-Valve-DDT-CD-12VDC-/290763981675
Relay SPDT Sealed	Sparkfun	COM-00100 ROHS	7	$1.95	$10.50	https://www.sparkfun.com/products/100
*Solder-able Breadboard - Large	Sparkfun	PRT-12699 ROHS	2	$8.95	$17.80	https://www.sparkfun.com/products/12699
*Arduino Uno - R3	Sparkfun	DEV-11021 ROH	1	$24.95	$24.95	https://www.sparkfun.com/products/11021
*Female Headers	--	--	7	--	--	--
Smartwater Bottle	Greenhouse Cafe	--	5	$3.50	$17.50	--
¾" Female to ½" Female PVC Threaded Adapter	Lowe's	435101RMC	5	$.60	$3.00	http://www.lowes.com/pd_126919-1815-435101RMC_0__
Hamilton Beach Wave Station Express Dispensing Blender	Amazon	54618	1	$19.01	$19.01	http://www.amazon.com/dp/B0065OK8AM/ref=pe_385040_121528360_TE_dp_1
Remote Control and Power Outlet	Amazon	025706341288	1	$8.33	$8.33	http://www.amazon.com/dp/B00FAII824/ref=pe_385040_127541860_TE_dp_1
*12V DC Power Supply	--	--	--	--	--	--

GRAND TOTAL: > $166.64 + the cost of the below.

We also used the following items found in the ES50 Lab:
Clear Tubing, ½" Acrylic, ¼" Acrylic, Solder, Hot Glue, Electrical Wires, Acrylic Weld, Duct Tape, Alligator Clips

For demonstration purposes we used the following:
Mocha Syrup, Caramel Syrup, Milk, Ice, Sugar, Water, Xanthan Gum, Cups, Coffee

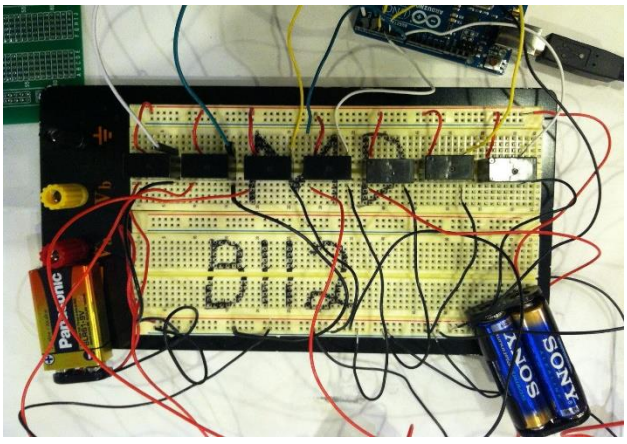
PROJECT IMPLEMENTATION

The following steps can be completed in any order:

(1)We designed the physical structure of the acrylic to be laser cut and we laser cut and glued all the acrylic together.
(2)We cut the bottoms of the Smartwater bottles, attached the PVC adaptors to the bottles, attached solenoids to the PVC adaptors, and attached tubing to the solenoids. We used hot glue and duct tape to try to seal the attachments.
(3)We opened the remote control and soldered two electrical wires each to both the on and the off buttons.
(4)We coded the online order form that would control the Arduino and send signals to each relay.
(5)We soldered the female headers to the breadboard and placed relays into the female headers. We soldered wires to the breadboard such that:

(a)each relay could be grounded in a common ground
(b)each relay could be connected to either a solenoid or to the buttons of the remote control
(c)each relay could be connected to a port in the Arduino
(d)each solenoid could be grounded in a common ground
(e)both of the buttons of the remote control could be grounded in a separate common ground
(f)the power supply could correctly supply power to the circuit to power the individual relays that controlled the solenoids
(g)the common grounds of both the relays that controlled the solenoids and the relays that controlled the remote control could be grounded in the ground ports of the Arduino

The following photo shows a prototype of a portion of the wiring on a non-solderable breadboard, before we switched to using the 12V DC power supply:



The indicated order should be observed for the remainder of the steps:

(6)We placed the bottles and tubing in their respective sockets of our assembled laser-cut structure. We also put the blender in place at the bottom of the structure.
(7)We connected the wires from the breadboard to the solenoids, and soldered them to the solenoids.
(8)We soldered the wires from remote control to the appropriate wires on the breadboard.
(9)We connected the wires from each relay to a port on the Arduino port, and we connected wires from both common grounds to a ground port on the Arduino.
(10)We connected the power supply to the circuit using alligator clips.
(11)We connected the Arduino to the computer to start testing our code. We made corrections and adjustments in our code to yield the perfect recipe!

In terms of practical advice, the biggest lessons that we learned centered around the importance of building a prototype circuit and testing the circuit every step of the way, the importance of grounding components in the proper common ground(s), the importance of soldering wires and keeping them as clean as possible immediately -- because often times a circuit simply does not perform because of a weak connection, and the importance of time management and careful planning before diving into construction.

A photo of our project before everything was wired together:

TEAM MANAGEMENT

Student B did much of the work on the front end by coding the entire arduino-web interface and building the relay-solenoid circuit prototype. Student A took apart the remote control and built the relay-remote-control circuit prototype. Student C put together the final circuits and soldered everything. Additionally, Student C designed and laser cut the physical structure of the machine, with significant input from Student A and Student B. Student A and Student C constructed the final physical product. Student B, Student A, and Student C debugged the circuits once the code was synced with the final product, and Student A and Student C tested and adjusted to taste the timing for all the dispensing and blending.

OUTLOOK AND POSSIBLE IMPROVEMENTS

With more time and resources, this project could be improved with the following:

- The project could include automatic dispensing into a cup. A solenoid could be attached to the spout of the blender and the solenoid would open as soon as the blender finishes its cycle. Furthermore, there could be some type of object detecting sensor so that the blender could not dispense unless an empty cup is in front of it.
- The project could have cleaner, more efficient, and less obstructive wiring. A case would be cut out of acrylic to house both the breadboard and the arduino.
- The project could have leak-proof connections and tubing. Attachments with the same threading as the bottles that we used could be designed and 3D printed, and silicone or some type of rubber stopper would be used to seal the tubing.
- The project could have the additional option of a vanilla frappuccino which, unlike the three flavors included in the original project, also takes a vanilla powder as an ingredient. So the project could include a tube with a trap door mechanism controlled by a servo in order to dispense an appropriate amount of vanilla powder into the blender.
- The project would be more user friendly in its ability to be cleaned. Perhaps a cleaning function would be implemented so that the user could run water through all the bottles and solenoids. Or the machine would be built with unscrewable, disposable, or easily replaceable components.

ACKNOWLEDGEMENTS:

Frapparduino would not have been possible without the help and support of the ES50 Team. We would like to thank:

Ivan Cisneros, our project TF, for meeting with us at a moment's notice whenever we needed him and for helping us become experts at laser cutting;
Jessica Lam, for all her optimism and support;
Jason Smith, for his great music playlists, general know-how, intentional name-forgetting, and sass which carried us through when we were at the end of our rope;
Joy Hui, for her determination in finding us relays at the last minute;
Xuan Liang, for her ever-present help in quickly ordering parts and for finding things in the lab for us to use;
Yasha Iravanthci and select members of the ES51 Team in Pierce, who either eagerly or reluctantly aided in our acrylic-acquisition efforts;
Chris Lombardo, for the 4:00 AM lesson on the importance of grounding properly the night before the fair when we were uncertain whether or not our project would come together;
Marko Loncar, for being involved with our project from its infancy and for proposing so many solutions, such as the remote control power outlet and the use of a 12V DC power supply.

DISCLAIMER

This report, code, and the included photos can be shared.

REFERENCES

We employed a substantial amount of trial-and-error, and in addition to those whom we thanked and the ES50 course material, we would like to give credit to:

- http://www.instructables.com/id/Build-A-Mobile-Bar-BaR2D2/
- http://www.instructables.com/id/Robotic-Drink-Mixer/
- The TiniBot ES50 Project Report
- The Smart Room ES50 Project Report

APPENDIX

Our code is linked in the design part of this report.