# AUTOKEGERATOR

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

Description: Implementation of an "automatic" kegerator capable of tracking amount poured, drinks remaining, temperature, etc.

By: Student A and Student B

Class: ES50
Instructor: Marko Loncar & Christopher Lombardo

December 2014

**ABSTRACT**

The AutoKegerator seeks to improve the kegerator experience. Using an Arduino in conjunction with a flowmeter and other basic components, the AutoKegerator is able to pour beer with a push of a button, then display on an LCD screen exactly how much beer was poured. Alternatively, the AutoKegerator will pour you exactly 12 ounces – the perfect beer. It also tracks the number of drinks remaining in the keg and the temperature inside the kegerator. We were motivated to complete this project to improve our personal kegerator for future use!

## INTRODUCTION

The AutoKegerator can be easily installed on any standard single tap kegerator. Upon installation, its features ensure that your kegerator experience will be streamlined and enhanced.

The system includes a two-line LCD screen for displaying output. There are two buttons for pouring: the Pour Button and the Auto Button. The Pour Button allows beer to pour as long as the button is pressed. Upon releasing the Pour Button, the screen displays the amount of beer that was just poured in ounces, so you know how much is in your cup. The Auto Button allows exactly 12 ounces of beer (one standard drink) to be poured each time it is pressed. Between pours, the screen displays the number of drinks (in 12-ounce servings) remaining in the keg as well as the temperature inside the kegerator.

To accomplish these features, the AutoKegerator uses an Arduino to communicate with the main components: an LCD screen, an in-line solenoid valve and flowmeter, and a temperature sensor. Fundamentally, the solenoid valve allows beer to flow or not to flow, and the flowmeter tracks the amount of beer that has been poured by sending the Arduino a pulse every time an internal turbine rotates (about once every 2.25 milliliters). The temperature sensor tracks the temperature inside the kegerator. The LCD screen displays the relevant information to the user. A more detailed analysis of the operating principles is provided throughout this report.

Once properly installed on a kegerator, the AutoKegerator system makes drinking beer from a kegerator easier and more fun, which is exactly why we chose to complete this project!

## DESIGN

Upon deciding to tackle this project, we had a number of design decisions to make.

The first decision we made was regarding temperature control. We knew we wanted to display the current temperature inside the kegerator, but we wondered if it would also be beneficial to control (increase or decrease) the temperature by pressing buttons on the AutoKegerator console. For most standard kegerators, including the one we were using, there is a knob inside the kegerator that can be used to change temperature. After researching the complexity of connecting to the kegerator's refrigeration component, and considering it was already quite easy to control the temperature before the AutoKegerator was installed, we decided not to implement temperature control, but rather just display the current temperature.

The next design decision we made regarded the number of buttons. At first, we thought it might be best to have four buttons, each corresponding to a different size pour (8 oz, 10 oz, 12 oz, etc). After deliberation we decided that a general "pour"

button would be needed – a button that poured as long as it was pressed. We also decided that the only automated pour should be the size of a standard beer, which is 12 ounces. Thus, we ended up with only the Pour Button and Auto Button.

While in the designing stages, we strongly considered adding a sensor to detect whether or not a cup was underneath the tap. We hoped that if the sensor indicated that there was no cup, then the system would not pour until a cup was detected. To accomplish this, we considered using a proximity sensor that would be placed either at the base or midway up the tap. However, as Professor Lombardo pointed out to us during our final project update presentation, we would likely have a problem with foam; to pour a beer correctly (and reduce foam), one must hold the cup within an inch of the tap at first, then move it farther away and at a different angle as the cup gets fuller. After testing possible pour tactics and the resulting amount of foam, we determined that a cup sensor would ultimately cause more false negatives than we would like, so we decided to not implement the sensor.

Perhaps our most important design decision regarded the power supply. Some of our components needed 12V (solenoid valve), some needed 5V (temperature sensor, LCD screen) and some could use either 12V or 5V (flowmeter). Additionally, the Arduino could be powered from either the Vin pin or the non-USB power jack with 7-12V. At first, we tried to have only a 12V supply, and create a voltage stepdown to 5V by using two resistors, $R_1$ and $R_2$, implemented in the following way:

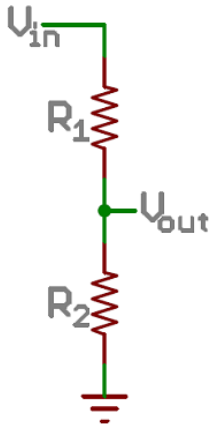


**Figure 1: 12V to 5V stepdown**

Using the equation $V_{out} = V_{in} \cdot \frac{R_2}{R_1+R_2}$, we set $R_1 = 1\text{K}\ \Omega$ and $R_2 = 750\ \Omega$, we were able to achieve $V_{out} \approx 5V$ from $V_{in} = 12V$. However, we soon realized that using this setup did not provide as consistent 5V supply as we would like. So, we created an elegant solution: we kept with only a 12V supply, and powered the necessary components with that, including the Arduino via the Vin pin. Then, we ran the Arduino's 5V output pin to a different rail on our circuit to provide consistent 5V power for the components that needed it. In essence, we used the Arduino itself as a way to get from 12V to 5V.

Our last major design decision addressed in which order to place the flowmeter and solenoid valve. We chose to put the flowmeter before the solenoid valve in the beer line; this way, the flowmeter would always have liquid beer running through it (never air or foam, as could be the case if the flowmeter was after the solenoid valve). This decision may have been inconsequential, but we think it helps the flowmeter give a more accurate reading.

Once all of these design decisions were made, the corresponding block diagram for the AutoKegerator system took form, as seen below.

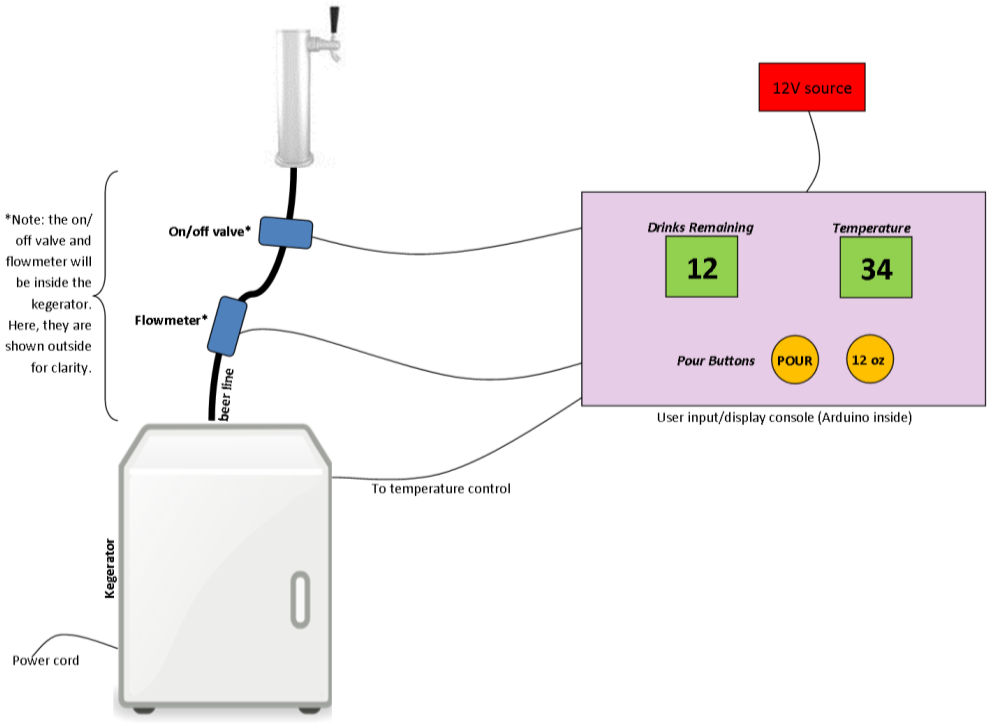


**Figure 2: Block diagram**

We were also able to produce the project's schematic. Most of the wiring was determined by looking at the recommended schematic for each individual component provided online, and then putting everything together. Of particular note is how the solenoid valve operates. Because it requires 12V to actuate, but the Arduino's HIGH output only gives 5V, we used a transistor to allow the 12V to flow to the solenoid only if 5V is applied to the transistor's gate. Due to the solenoid's relatively high current (1 amp), it took trial and error to find an appropriate transistor and diode capable of completing the task. The project's schematic is shown below.

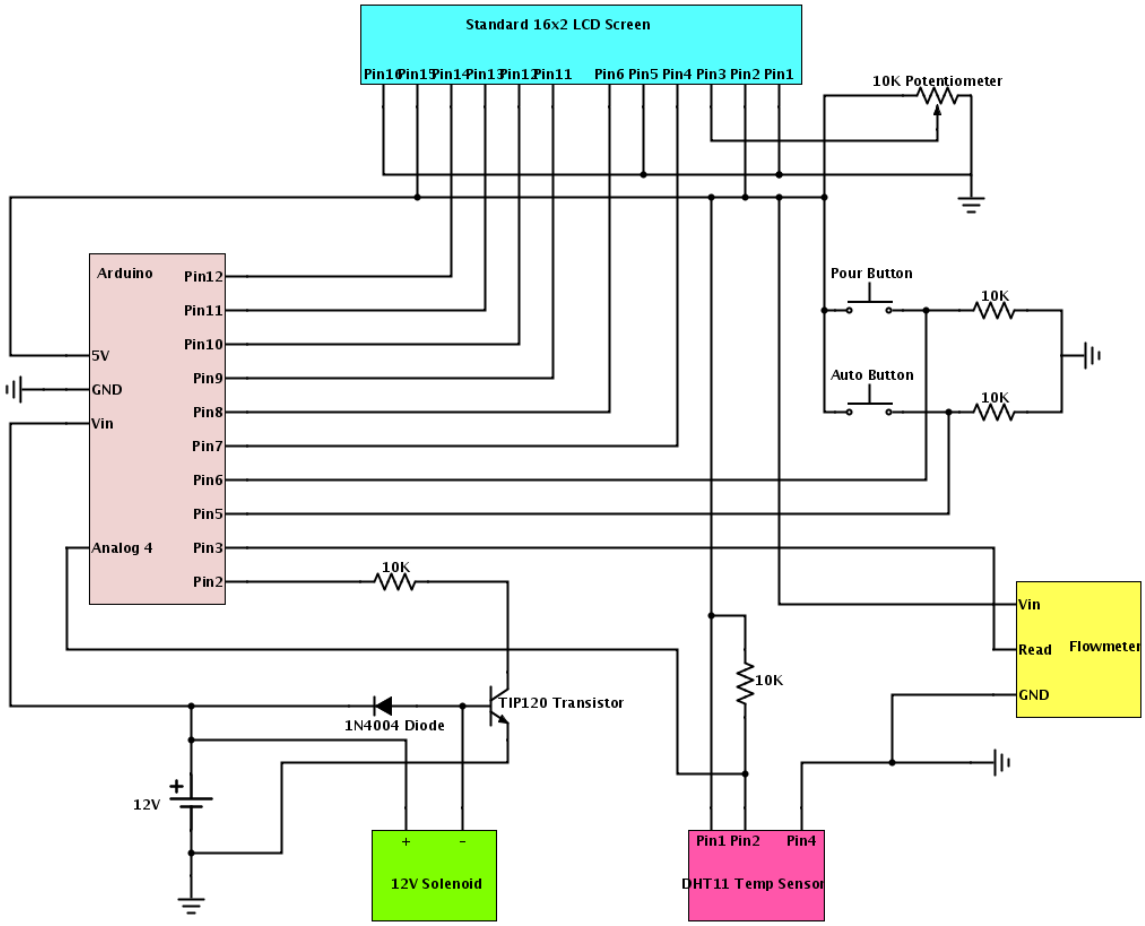


Figure 3: Schematic

For testing purposes, we first implemented our system on a breadboard in the lab, with the solenoid valve and flowmeter not connected to the kegerator beer line. We were able to blow into the flowmeter to receive pulses from it. Thus, we were not able to test pouring of exact quantities of liquid, but we were able to make sure the system was generally working. The photos below show the prototyping process. Note that there are four buttons (rather than two) in the pictures, as we were still deciding exactly how many buttons to use.

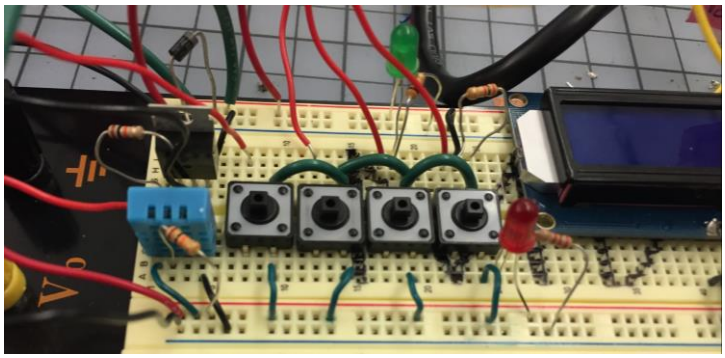

Figure 4: Left side of breadboard

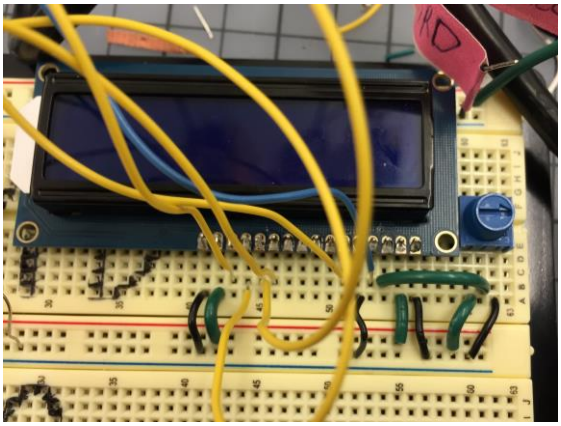

Figure 5: Right side of breadboard

Figure 6: Flowmeter (left) and solenoid valve (right)

## PARTS LIST

The table below summarizes all parts and materials that we used.

| Part | Quantity | Comments | Part number or Link | Source | Unit Price | Total Price |
|---|---|---|---|---|---|---|
| Arduino Uno | 1 | From lab. | 50 | Adafruit | 24.95 | 24.95 |
| Liquid Flow Meter - Plastic 1/2" NPS Threaded | 1 | | 828 | Adafruit | 9.95 | 9.95 |
| DC 12V 1/4 Inch Electric Solenoid Valve | 1 | | http://www.amazon.com/dp/B00827FP26/ref=pe_385040_127541860_TE_dp_i1 | Amazon | 11.29 | 11.29 |
| DHT11 Basic Temperature Sensor | 1 | | 386 | Adafruit | 5.00 | 5.00 |
| Standard LCD 16x2 | 1 | | 181 | Adafruit | 9.95 | 9.95 |
| Pushbutton | 2 | From lab. The ones references from Adafruit are not exactly the same. | 2093 | Adafruit | 4.95 | 9.90 |
| 10k Ohm Resistor | 4 | From lab. | http://www.amazon.com/E-Projects-10k-Resistors-Watt-Pieces/dp/B00BWYS9BA/ref=sr_1_1?ie=UTF8&qid=1418497886&sr=8-1&keywords=10k+resistors | Amazon | 0.05 | 0.20 |
| 1N4004 Rectifier Diode | 1 | From lab. | 35991 | Jameco | 0.05 | 0.05 |
| TIP120 Transistor | 1 | From lab. | 32993 | Jameco | 0.39 | 0.39 |
| AA Batteries | 8 | From lab. | http://www.amazon.com/Duracell-Coppertop-Batteries-MN1500-Alkaline/dp/B003VSCLMI/ref=sr_1_5?s=hpc&ie=UTF8&qid=1418498353&sr=1-5&keywords=aa+batteries | Amazon | 0.50 | 4.00 |
| Battery Holder | 1 | From lab. | http://www.amazon.com/Battery-Holder-Standard-Snap-Connector/dp/B000LFRTIK/ref=sr_1_1?s=hpc&ie=UTF8&qid=1418498428&sr=1-1&keywords=aa+batteries+holder | Amazon | 5.79 | 5.79 |
| Silicone Cover Stranded-Core Wire | 1 | From lab. | 1970 | Adafruit | 1.90 | 1.90 |
| Circuit Board | 1 | From lab. | http://www.radioshack.com/radioshack-printed-circuit-board/2760170.html#start=7 | Radio Shack | 3.49 | 3.49 |
| 1/4" Hose Clamp | 2 | Used to connect flowmeter/solenoid into tubing. | http://www.homedepot.com/p/Everbilt-3-8-7-8-in-Hose-Repair-Clamp-6706595/100198182 | Home Depot | 0.94 | 1.88 |
| 1/4" Hose Barb x 1/4" NPT | 2 | Used to connect flowmeter/solenoid into tubing. | http://www.amazon.com/Anderson-Metals-57001-Fitting-Adapter/dp/B002SAO7XQ/ref=sr_1_1?ie=UTF8&qid=1418499027&sr=8-1&keywords=1%2F4%22+hose+barb | Home Depot | 3.07 | 6.14 |
| Copper Pieces to Connect Flowmeter and Solenoid | 1 | Will vary depending on what is available at local harware store. Price is estimation. | NA | Home Depot | 5.00 | 5.00 |
| Pipe Joint Compound | 1 | Used on fittings to prevent leaks. | http://www.homedepot.com/p/Oatey-8-oz-Pipe-Joint-Compound-154202/100204007 | Home Depot | 3.34 | 3.34 |
| | | | | | **Total Cost:** | **$ 103.22** |

**Figure 7: Parts list**

## PROJECT IMPLEMENTATION

*Step One: Skills Required*
Knowledge of basic electronics is helpful, but this guide should be enough for even the most novice hobbyist to complete the project. Some knowledge of computer code (C++) may be needed if you wish to alter the program that is loaded onto the Arduino.

*Step Two: Tools and Materials*
Having access to an electronics lab is encouraged, as we did. Wire strippers, DC voltage sources, soldering equipment and a multimeter are very helpful for building and debugging. For parts, see the complete list above (Figure 7).

*Step Three: Make a Prototype on a Breadboard*
We first implemented the AutoKegerator on a breadboard by wiring everything as shown in the schematic (see Figure 3 for the schematic, and Figures 4 and 5 for our breadboard). Rather than connect the AA batteries as your 12V source right now, it is highly encouraged to use a DC power source, so you can quickly and easily disconnect the voltage if things go awry. We uploaded simple codes to the Arduino to test each component of our new prototype.

We had several issues arise during this phase – we will cover them here so they don't happen to you. First, we had issues with the solenoid due to the high current required. We had to test several different transistors and diodes, as well as different configurations, until we found something that worked. We quickly discovered that a simple light emitting diode (LED) could not handle the current, so we had to upgrade to the 1N4004. Similarly, the TIP120 is the only transistor that we found that did not burn up after a few seconds of the voltage being turned on.

Our second main issue came from the buttons. At first, we did not include the 10K Ohm resistors to ground. Without these resistors, the buttons consistently registered false positives due to the Arduino acting as a capacitor and sending current back to the buttons. Once this was discovered and the resistors were added to allow stray voltage to dissipate, the buttons operated properly.

*Step Four: Upload AutoKeg.ino*
We next uploaded the actual code that would run the system, and tested it. With the solenoid and buttons already functioning properly, the main goal was to make sure that the LCD screen was wired correctly, and that the two buttons were communicating with the Arduino as specified. The Pour Button was easy to test despite the fact that the system was not actually attached to a kegerator yet – when pressed, we could hear the solenoid valve open, and when released we could hear the solenoid valve close. The Auto Button was a bit trickier. We tested it by pressing and releasing the button, waiting for the solenoid to open, and then blowing air through the flowmeter until the solenoid closed. In this way, we were able to see if the general framework worked correctly, but not that the exact amount of liquid was triggering the closing of the solenoid. For that, we would have to wait until the system was actually installed on a kegerator.

In all, it took us several hours to debug AutoKeg.ino, but we did finally get everything working properly. For you, it should suffice to simply upload the program – it should work correctly right out of the box. To be safe, you should perform the button tests as described above to make sure everything seems right.

*Step Five: Connect the Flowmeter and Solenoid Valve*
At this point, you should have everything wired correctly, and the LCD screen working. The flowmeter and solenoid valve should be connected to the breadboard and Arduino by five wires, but they should not be connected to each other. At this point, use the fittings picked up at a hardware store to screw the flowmeter and solenoid together. Make sure that the arrows on each are pointing in the same direction, and that the flowmeter is before the solenoid valve. The hose barbs go at each end so that you can insert it into the hose later on. It may be easier to disconnect the wires to accomplish this; feel free to do so, just be sure that you connect them again in the same way. We recommend using a pipe joint sealant on each connection to make sure you don't have any leaks. Once fully connected, it should look something like Figure 8 below.

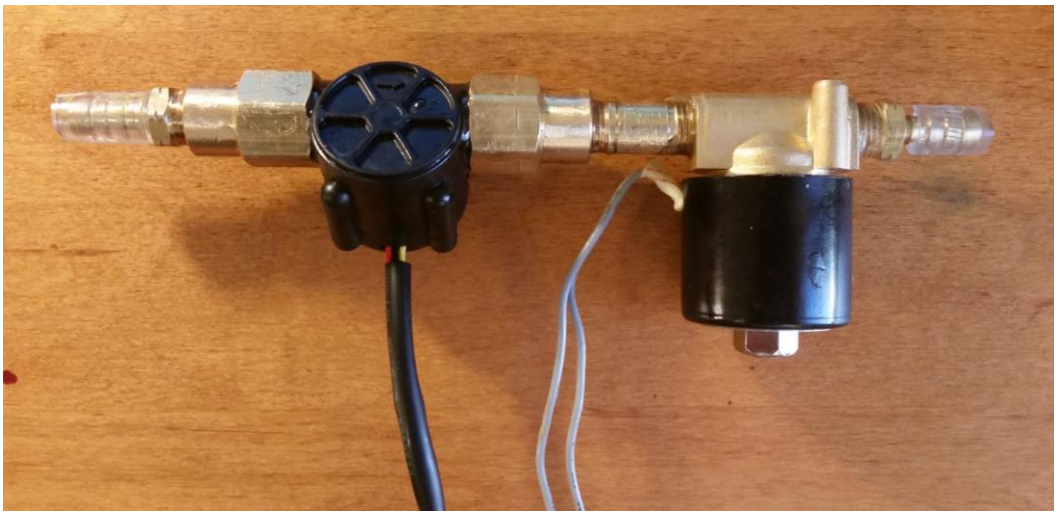

**Figure 8: Flowmeter/solenoid connection**

*Step Six: Solder to the Circuit Board*
At this point, you are ready to make the prototype permanent by soldering it to the circuit board. Carefully move each component from the breadboard to the circuit board and solder into place. Once complete, redo Step Four (testing) to make sure everything is working properly. After we soldered, we had a couple loose connections that we had to go back to and solder again. See below for a picture of our completed circuit board.

Figure 9: Circuit board

*Step Seven: Install into Kegerator*
Before you begin, be sure to fully disconnect everything in the kegerator, including the power cord, CO2 tank, and keg. Once that is done, look for the beer hose that goes from the keg to the top of the kegerator (and out to the faucet). About 4 inches below where the hose exists the kegerator, make a clean cut to the hose with scissors. At this point, you need to install the flowmeter/solenoid into each end of the cut, thus reconnecting the hose. Use the hose clamps to secure the tubing into place. To accomplish this, we had to cut the 5 wires (positive/negative to the solenoid; positive/negative/read to the flowmeter) that go to the flowmeter/solenoid. After everything was in place, we ran those 5 wires up through the hole that the faucet uses, and reconnected them by bending them and using electrical tape.

Lastly, you should drop the temperature sensor through the same hole, and secure it with tape to an inside wall of the kegerator. Unfortunately for us, our temperature sensor turned out to be faulty, so we just left it on the outside. We have a new sensor ordered, though, and plan to install it when it arrives.

The picture below shows us in the process of installing the AutoKegerator system.


Figure 10: Installing into kegerator

*Step Eight: Test and Calibrate*
Once everything is connected, we set the circuit board, Arduino and battery pack on top of the kegerator. Now is time to test and calibrate. To start, push the Pour Button for a few seconds (with a container under the faucet to catch the beer), and observe each component to ensure there are no leaks. If the system is leak-free, get a new container and press the Auto Button. After the foam has settled, measure the amount of beer (or look at the LCD screen) to see how much comes out. For us, it was pouring less than 12 ounces at first, but once we debugged the Arduino code a little more, we got it to the correct amount.

A word of caution: we found that using the Auto Button should only be advised if the keg is properly pressurized and has not been shaken. The system is designed to dispense 12 ounces of liquid beer; if the pressure isn't quite right, beer tends to foam considerably as it comes out of the faucet. Since foam has much more volume than liquid beer, it is very possible to overflow a glass.

*Step Nine: Enjoy!*
The system works! See below for pictures of the final product, and check out the link below for a video of the AutoKegerator in action.

Video link: https://www.youtube.com/watch?v=BdFVvPOHl9k&feature=youtu.be

Pictures:

**Figure 11: Inside of the kegerator**

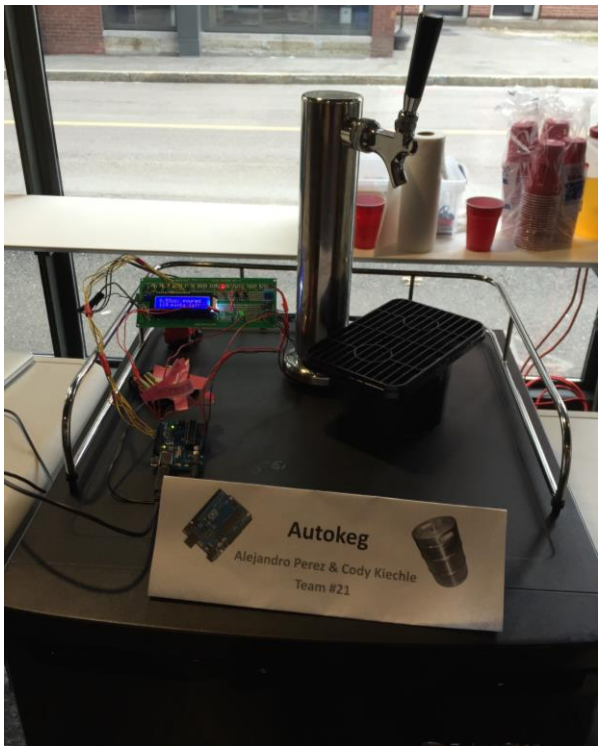

Figure 12: the AutoKegerator

## TEAM MANAGEMENT

We worked together quite well while building this project. Throughout the planning stages, we were in regular communication about potential ideas for the system. When we started to build, Student A focused on the breadboard and Student B focused on writing the code. We both helped solder to the circuit board and install in the actual kegerator. Lastly, we were both involved in the final testing and calibrating.

## OUTLOOK AND POSSIBLE IMPROVEMENTS

We believe the most helpful improvement would be to create a more compact, rugged, and aesthetically pleasing version of the system. Specifically, we would like to use the laser cutter to create an enclosure with only the buttons and LCD screen being displayed. The Arduino and circuit board can be safely stored inside the enclosure.

We also might like to install a light that flashes or makes a noise when the keg is nearing empty – so you always know when it's time to buy a new keg.

**ACKNOWLEDGEMENTS**:

We would like to thank the entire ES50 team, but most notably Professor Loncar and Professor Lombardo for staying late in lab on multiple nights to help us debug the system. We would also like to thank Willie Pirc, who provided great advice throughout the process.

**DISCLAIMER**

This report and all its contents, in addition to the Arduino code, can be shared freely.

**REFERENCES:**

The following websites provided inspiration and helped with some of the technical components of our project:

http://kegbooty.com/Cheap_Keg_Refrigerators.html

http://kegduino.org/

https://www.youtube.com/watch?v=GWUYBrEKLRU

We also referenced Micahel Margolis' *Arduino Cookbook* (2011) on several occasions.

**APPENDIX**

Please see the attached AutoKeg.ino for the code used to program the Arduino.