2013

EECE 281 B8

Andy Au
Daniel Chong
Gregory Lee
Jae Yeong Bae
Jeffrey Chan
Red (Kyle) Garsuta

# AUTOMATIC SODA FOUNTAIN DISPENSER

# Contents

# 1 Introduction

## 1.1 Specifications

The specifications were to design, build, and test an automatic soda fountain dispenser. The soda dispenser must be implemented on the Altera DE2 FPGA using a DE2-8052 soft-processor. The requirements are as follows. [1]

1. **Programmed in assembly language.** The controller software must be written in assembly language. The user is allowed to select the amount of liquid he/she wants dispensed as a percentage of a full cup (i.e. select 75% for a ¾ full cup).

2. **Capacitive height liquid measurement.** A simple capacitor can be built by attaching two pieces of aluminum tape to the sides of a plastic cup. The capacitance of which is proportional to the dielectric between the plates. By measuring the capacitance, the liquid amount can be determined and the controller programmed to react accordingly.

3. **Servo based soda valve.** By programmatically opening and closing a servo valve, the amount of liquid poured in the cup can be controlled.

4. **Temperature display.** The water temperature, in degrees Celsius, is indicated on the DE2's 7-segment displays. For this purpose, the LM335 temperature sensor is used.

## 1.2 Overview and Design Approach

A thing is perfect not when nothing more can be added to it, but when nothing more can be taken away. With this idea in mind, we sought to make our project as compact and efficient as possible. We started by estimating the size of the main circuit boards to determine the overall dimensions of the project.

Once this was done, we created 3D models of the project using AutoCAD and Google Sketchup, and agreed upon the general appearance of the soda dispenser. The team was split into two main groups of three members, each group in charge of software or hardware.

Once the project functionality was decided, each group worked simultaneously while coordinating project developments to assure overall compatibility. Figure 1 shows an overview of the soda dispenser's implementation. Section 2 of this report delves in to the design process, while detailed descriptions of the software and hardware implementations are described in Section 3.

[1] Jesus Calvino-Fraga, *Project 1: Automatic Soda Fountain Dispenser* (University of British Columbia, 2013), p.1.

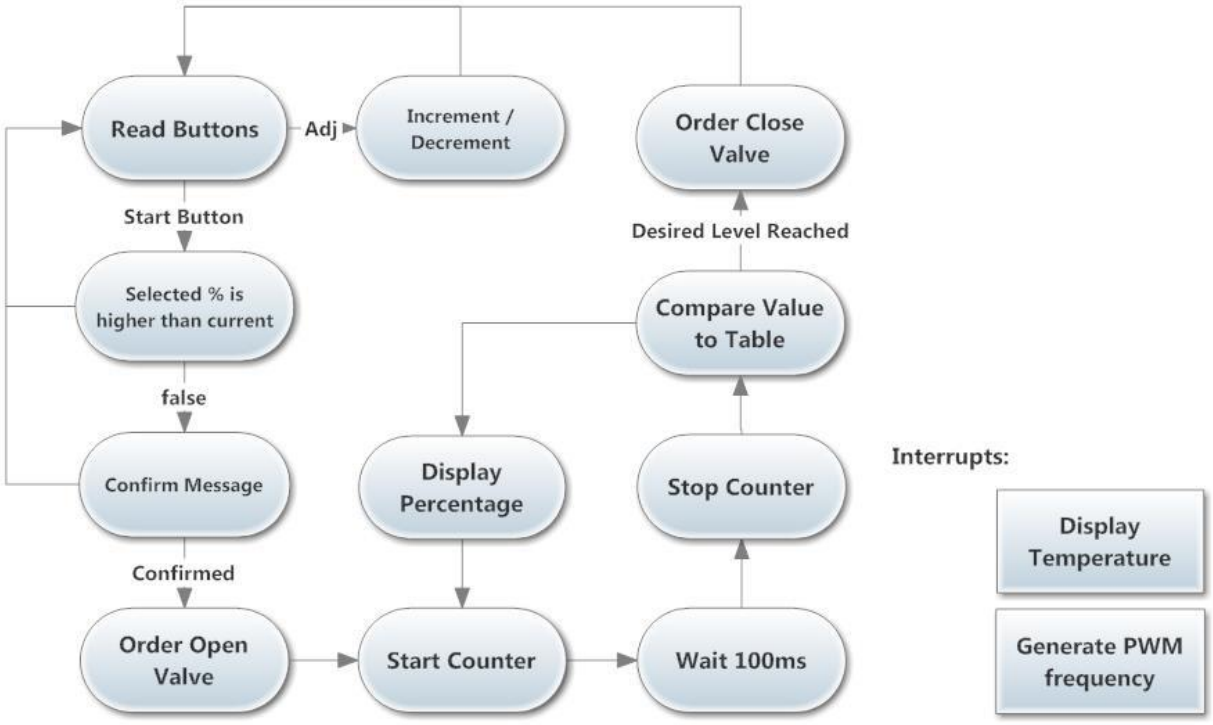


**Figure 1: Soda dispenser implementation overview**

# 2 Investigation

## 2.1 Idea Generation

The idea for the project was generated through numerous group discussions and activities. To begin with, we met bi-weekly for two hours to discuss the project. Initially, we started out by first laying out what we wanted the project to be like, without limitations. This helped us create a foundation that we could refer to. Next, we laid out what we knew and what we could do to accommodate our ideas. This included a capacitance meter for the water levelling and frequency control in assembly for the servo motor. Once these two elements were discussed thoroughly, we were able to come up with a realistic idea of the result of our project

## 2.2 Investigation Design

In order to get the investigation of the design done efficiently, we spread out research to be done within the group. Before each meeting, we would come up with an idea and support as to why this idea could work. Once a realistic final design was decided upon, we began to work on the actual production of our project. Of course, we ran into obstacles to which we had to work around from in our original design. Some examples include angle of which to turn the servo motor to determine the desired flow rate, determining the strength of the motor for our valve, and

also how to fit the circuitry into a compact shell. Most of the information was found either from previous laboratory exercises or done through trial and error.

## 2.3 Data Collection

To ensure that our dispenser outputs the correct amount of water, we had to first ensure the correct operation of the servomotor. Since the servo motor is controlled by the pulse width, we connected the output from our circuitry (which is supposed to be connected to the servomotor) into the oscilloscope to ensure that our code was generating the desired signals with correct pulse widths.

After we collected data for the servo motor, we tested to what degree the valve should be opened for an optimal output water flow; we started at turning the valve 45° and moved incrementally upwards, measuring the time it takes for the cup to be 100% full.

Furthermore, in order to measure the water level actively, we have to calibrate our program to the capacitance at different water levels. Instead of using an oscilloscope, we programmed our DE2 board to display the capacitance when debugger mode was activated. We first took the value of the capacitance when the cup was empty, recorded the values into excel, and increased the water level at measured increments; this allowed us to draw a calibration curve and thus we could program this into our code.

## 2.4 Data Synthesis

After trial and error with different methods of pulse width modulation, we concluded that it is the most efficient to control the servo motor by the use of an interrupt, to which the position of the servo is controlled by the frequency being sent to it. Through measuring the time required for the water to reach the desired levels, we came to the conclusion that it is optimal to open the valve fully (valve turning 90°). By plotting the calibration curve on excel, we realised the capacitance and water level has revealed a logarithmic relationship, and concluded to use a lookup table for conversion between the two variables, as we only require the conversion at certain values, and the other parts can be approximated linearly.

## 2.5 Analysis of Results

Research into pulse width modulation revealed that it was the most efficient to modulate digitally, due to analogue pulse width modulation circuitry being technically complex to construct and also requires a plethora amount time to construct. Learning from our previous projects, we concluded that it was inefficient to do pulse width modulation through analogue circuitry.

As the time taken to close the valve is less than 0.6 seconds, we concluded that the delay between reaching the water level and closing of the valve was within tolerable bounds, and the overflow of water caused by the delay was relatively negligible.

Due to the limitations of programming in 8052 assembly language being unable to directly calculate logarithmic values, we concluded that it would be unnecessary to use Pade approximation or Taylor series approximation for the calibration curve, and a lookup table and linear approximation between the lookup values was appropriate.

Contrary to capacitance, the calibration curve of the temperature sensor was linear, and we concluded there are less degrees of error if the curve was directly programmed.

# 3 Design

## 3.1 Use of Process

Being in a team of six members with a software and hardware component to complete, we split our group into two sections with the main focus of each group on either hardware or software. Within these groups, we designed our project using modularity in such a way that each member could work on a separate component. By being conscious of the overall team design and any changes made to it, this allowed for the majority of team members to be constantly contributing to the project. Using modularity for the basis of our project design would also allow us to debug problems separately to reduce the time spent analyzing problems.

To improve the design process and give our team a visual representation of the project, we used AutoCAD to build models of any hardware designs we had. Using such software allowed us to make visual changes quickly and allowed any member to view the changes immediately.

## 3.2 Need and Constraint Identification

For this project, practicality and functionality were our number one priorities.

1. **Portability of the Cup.** From a user point of view, the cup could not be attached to the water dispenser at all. Otherwise, there would be no way to actually drink the water.

2. **Portability of the Dispenser.** Our group came up with the idea that the water dispenser should small and compact. This would allow the dispenser to be placed almost anywhere by the user which is important especially when space is tight.

3. **Emergency Stop.** In our project, we had two emergency stop features implemented. One of them is an actual button where pressing the button would stop the dispensing of water. This would be useful if the user accidentally chose too much water. Our second emergency stop occurs when the cup is touched or removed while water is still being poured. Again, water would immediately stop dispensing.

4. **Water Level Increments.** For our water dispenser, we chose to go with 20% increments, starting from 0% and going up to 100%. While we could have made the increments

smaller, the difference between 30% and 40% is so small compared to the size of the cup that it would be redundant to do so.

## 3.3 Problem Specification

To meet the constraints and needs our group identified, we had to tweak and change our design slightly.

1. **Portability of the Cup.** To meet this requirement, we had to figure out a way to attach leads to the water dispenser. Doing so in an elegant way proved to not be as trivial as we had initially thought.

2. **Portability of the Dispenser.** One of our original requirements was to build a water dispenser that was not too big. This made the portability of the dispenser easy as our design was already compact to begin with. In addition, the size of the cup had to be limited as too big of a cup would not fit in our dispenser.

3. **Emergency Stop.** An additional requirement for this project was that a fourth button had to be implemented. The secondary emergency stop feature forced us to write some assembly code that would turn off the water valve if a big jump in the value of the capacitance is read.

4. **Water Level Increments.** This feature did not change how we designed or made the water dispenser in any way. That is because changing how much the water increments is as simple as changing certain values in the assembly code.

## 3.4 Solution Generation

Based on the needs and constraints we identified as a group, we had to change and alter the design of our water dispenser. This did not drastically change how our project was originally planned but definitely made it slightly more challenging.

1. **Portability of the Cup.** To implement this into our design, we had to figure out where else we would attach the two leads responsible for measuring the capacitance of the cup. We decided to attach these leads to a tray that the cup was going to sit on when being filled up with water. As long as the leads touched the cup when it is placed on the tray, the capacitance could be measured, allowing us to determine the water level of the cup.

   We were originally not going to have a tray. Instead, we were going to attach the leads to the plastic cover where the cup would then sit on top of. The tray was added in later as it did not look as nice cosmetically.

2. **Portability of the Dispenser.** Our original design was already quite small to begin with. All that was required for us was to tweak with the measurements of our water dispenser for us to be happy with the new slightly more compact design. Figure 2 shows our modified preliminary design.

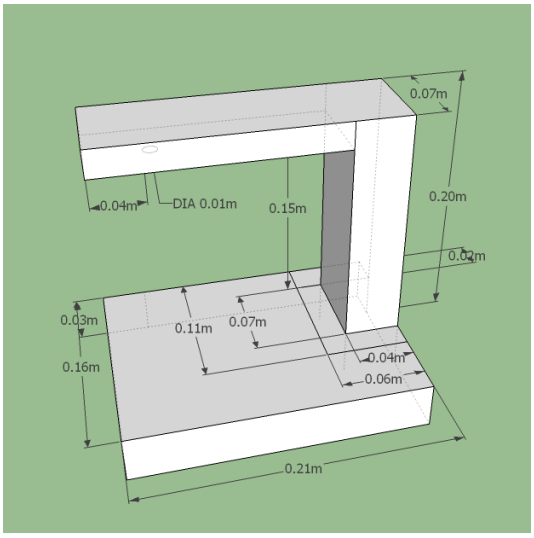


**Figure 2: Soda Dispenser Preliminary Design**

3. **Emergency Stop.** To implement the button, we just added a fourth button to our circuit board. The secondary emergency stop was completely software based. As soon as there was a drastic capacitance change, the servo would turn, closing the water valve. This quick change in capacitance corresponds to the cup being touched or removed in real life.

4. **Water Level Increments.** Choosing the increments that we wanted in our water dispenser was purely software based on decided on having the water level increment by 20%. While it could be argued that using smaller increments would give the user more control in how much water they want, the difference between a cup filled 65% of the way and a cup filled 60% of the way is negligible. In addition, having bigger increments simply made the testing stage of our project that much easier to do.

## 3.5 Solution Evaluation

1. **Portability of the Cup.** Implementing a metal tray with a plastic mesh cover and wires connecting to where the cup would be placed proved to be an easy solution to this problem. The only end requirements were to design the tray and to create a large enough surface area using wires for the cup to make proper contact with them. The tray was designed so that the combined height of the tray and the cup were just the right size for it to fit underneath the dispenser. This meant that adding a tray for aesthetic purposes as well as to catch water was a great improvement over leaving the tray out and did not hinder the portability of the cup.

2. **Portability of the Dispenser.** Since we used the Altera DE2 board for our project, our project required mostly software to control the dispenser. This meant that our small design had minimal impact on the effectiveness of the dispenser. The area required for the Altera DE2 board gave us more than enough room to implement any other hardware which is why there was no need to make the frame any larger than necessary. The end design allowed for easy removal of the tank by simply unscrewing the tank off the top of the dispenser to make the dispenser even more portable.

3. **Emergency Stop.** Requiring only the need of minor software additions as well as the implementation of a fourth button, the emergency stop feature was a practical feature to include in our project. This solution, when paired with the metal tray, prevented any possible spills from occurring. The user could also use the emergency stop if they desired a different amount than they initially desired.

4. **Water Level Increments.** By using 20% water level increments, we reduced testing time while keeping a reasonable range of testable water levels. This is because 20% increments are easy to visually inspect as well as only require several tests to determine the full range from the minimum 20% to full had no problems. There was also no problem with choosing 20% because changing the water level increment to a lower amount such as 10% would not be too difficult to implement as it only required a change in software.

## 3.6 Detailed Design

1. **Liquid Capacitance Measurement.** One of the most important block of this project was the liquid capacitance measurement. This block was used to measure the amount of water inside the cup, so we can dispense correct volume of water. We used an astable timer circuit, connecting the cup in place of controlling capacitor. As the cup acts as a capacitor with varying values from 0pF to 200pF, the timer circuit generates different frequencies depending on the amount of water inside the cup. We then used the assembly counter logic to capture the number of "ticks" given off by the timer in a predefined short amount of time. At this point, the actual value of capacitance is not important, but the correct mapping of number of ticks to percentage value of water inside the cup is. We arranged the time so that maximum number of ticks received during any interval is less than 256 so it can be calculated easily with an 8 bit register. We made a chart of these values compared to percentage of water for every 5% difference, and used array-like coding style (See Appendix A) to map to required value. While dispensing, this table is used to compute current percentage of water for display. This value is stored in register R4 to be displayed in parallel with temperature.

2. **Temperature Sensor.** The temperature sensor was a completely standalone feature on our project. It ran on interrupts regardless of other functions. We used the LM335 chip connected as in figure 3. According to the datasheet, the voltage read at output directly corresponds to actual temperature at 10mV/Kelvin. This terminal was connected to a successive approximation ADC using R-2R ladder DAC and a LM393 comparator chip. Every second, the processor runs this successive approximation to determine the value and computes the Celsius equivalent with provided math32 library. this value is stored in register R3 to be displayed in parallel with percentage value.

3. **Servo Motor Driver.** The DE2 board must constantly generate PWM signal in order to control the servo motor, whether it is closed or open. The timer interrupt is used to generate this PWM signal. Depending on value stored in register R7, the timer interrupt generates one of two PWM signals, controlling the servo position. Every time the ISR is ran, the logic determines the time required until next ISR call, and sets/clears the bit used to generate the signal.

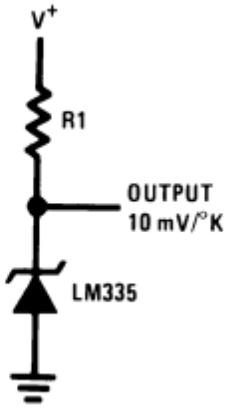


Figure 3: Temperature sensor diagram

4. **Display Driver.** The Display Driver is required in order to display two values, the water percentage and the temperature on the 7 segment display at same time. The role of this driver is to compute a number out of these two numbers, and display the number on the 7 segment display. The temperature value is multiplied by 10,000 in order to "shift" the number left by 4 digits, making room for the water percentage. Then the water percentage value is added to this number, to form one number including both temperature and the percentage. This value is then displayed onto the 7 segment display using provided driver.

## 3.7 Solution Assessment

1. **Liquid Capacitance Measurement.** We started my measuring actual capacitance of the cup at each 25% interval to determine a rough relationship. We found out that the capacitance and percentage formed a non-linear, but rather a logarithmic curve. Since it was much difficult and not worth it to emulate a log function within assembly, we chose to make a table of values at each 5% interval and refer to this table to determine current percentage. Then the actual capacitance of the cup was no longer important, but a correct value from counter matched to each percentage was. We designed a simple program to show the real-time value received from the timer to read off and make the table with. A weakness of our design from this portion was that as the percentage increased, detected value was less accurate and fluctuated more. This is due to the logarithmic nature of the function and could not be fixed. Difference between nearest values at 20% was as big as 100, whereas it was as small as 1 or 2 near 100%.

2. **Temperature Sensor.** In order to test the temperature sensor, we used a thermometer to compare the values. According to the datasheet, we did not need to calibrate the sensor and it was guaranteed to be accurate within 1 degree Kelvin. We tested this by resting a thermometer right next to the temperature sensor and confirmed that the readings matched. We tested this under normal condition and with hot air blowing through both sensors.

3. **Servo Motor Driver.** In order to test the servo motor, we used an Arduino to program a simple testing environment. Using a potentiometer, we could turn the servo to any desired position. We used the oscilloscope to read the required values at desired position and implemented the same signal in DE2. We programmed a simple code to test the servo at on/off position using a switch on DE2 board.

# 4 Life-long Learning

## 4.1 Software

There were several knowledge gaps during the project. In particular, the most significant one with relation to software was understanding and controlling the servo motor. Although we had done a lab exercise that helped us understand how to produce square waves with a desired frequency, it did not describe how this would help control the servo motor. After some research done on the motor itself and on pulse width modulation, we were able to solve how the servo worked using a potentiometer which controlled rotation of the motor.

The foundations of Assembly language that we learned in EECE 259 proved indispensable for this project. Basic understanding and experience with electrical components gained from EECE 280 was also invaluable.

## 4.2 Hardware

The compact hardware design required all members of the group to have a good understanding of the construction process. All six member received training in the following areas:

- Bending
- Drill Press
- Finishing
- Shearing
- Spot Welding

The three members in charge of hardware went on to learn the basics of 3D modelling software to aid the design process, namely AutoCAD Electrical and Google Sketchup. DraWinG (.dwg) files generated by the aforementioned software were also necessary in cutting material using the OMAX Waterjet Cutter available in the machine shop.

Of particular note was the difficulty of achieving precise results, which was of absolute importance given the scale of our design (i.e. it was very easy to bend a sheet of metal along a line which was off by a few degrees, resulting in incompatibilities, or drilling a hole a few millimetres off). Nevertheless, satisfactory results were achieved.

Aside from the standard machine shop training mentioned above (Bending, Drill Press, etc.), no further training is made available by the machine shop. We believe that further training in other fields could be of great benefit in future projects. For example, we started building projects using metals of certain thicknesses that we did not know anything about. This was a noticeable knowledge gap as the shop technicians obviously assumed that most of us were knowledgeable on these areas. For the most part, we did not know most of the materials/parts that were available to us. This knowledge would have been of great significance had it been available during the preliminary design process (i.e. we originally designed to use bolts/nuts, not knowing we could use rivets).

# 5 Conclusions

The initial goals and specifications were accomplished, and the project performs as expected. As such, this project was an overall success. Furthermore, the aesthetics of the final product are quite admirable due in part to the successful implementation of a compact design. Appendix B shows a snapshot of the completed project.

The team was able to effectively design and implement a hardware structure that sufficiently housed all the components, while eliminating any unnecessary space. Despite the successful result, this goal of compactness did create hurdles in the design process.

Due to the compact design, there was very little room for adjustment and minor discrepancies in component implementations resulted in reconstruction of incompatible parts (i.e. screw hole connection off by a few millimetres). The space allocated for circuitry was also very compact that careful and precise soldering methods were required. Amidst all these challenges, the team successfully followed through and met initial goals within time and scope.

# 6 Bibliography

Altera Corporation. *Development and Education Board User Manual.* (2006).

Arduino. *Servo Library*. Retrieved February 3, 2013, from http://arduino.cc/en/Tutorial/Knob

Calvino-Fraga, Jesus. *DE2-8052 Soft-Processor: Getting Started Guide.* (Vancouver: University of British Columbia, 2013).

Calvino-Fraga, Jesus. *Debugging Assembly Programs with Cmon51 in DE1-8052 and DE2-8052 Processors.* (Vancouver: University of British Columbia, 2013).

Calvino-Fraga, Jesus. *Module 1 - Digital to Analog and Analog to Digital Converters.* (Vancouver: University of British Columbia, 2013).

Calvino-Fraga, Jesus. *Module 2 - 555 Timer.* (Vancouver: University of British Columbia, 2013).

Calvino-Fraga, Jesus. *Module 3 - Interrupts.* (Vancouver: University of British Columbia, 2013).

Calvino-Fraga, Jesus. *Project 1: Automatic Soda Fountain Dispenser.* (Vancouver: University of British Columbia, 2013).

Intel Corporation. *Article Reprint AR-526: Analog/Digital Processing with Microcontrollers.* (Chandler).

MIT FabLab. *A Gentle Introduction to Abrasive (Waterjet) Machining.* Retrieved January 27, 2013, from http://rpl.mit.edu/tools/tutorials/Waterjet-OMax-1/A_Gentle_Introduction_to_Abrasive_Machining_-_Part_I.pdf.

Molex. *Capacitive Fluid Level Sensors.* Retrieved on January 27, 2013, from http://www.molex.com/molex/products/family?key=capacitive_fluid_level_sensors&channel=products&pageTitle=Introduction#overview.

Vault Information Services. *8051/8052 Instruction Set.* Retrieved February 3, 2013, from http://www.8052.com/set8051.

# 7 References

Calvino-Fraga, Jesus. *Project 1: Automatic Soda Fountain Dispenser.* (Vancouver: University of British Columbia, 2013).

# 8 Appendices

## 8.1 Appendix A: Sample Liquid Capacitance Code

```
;*************************************************************
; THIS CODE LOOPS THROUGH TABLE TO FIND CURRENT PERCENTAGE
;*************************************************************
     mov  dptr, #Cap_Values
     mov  R0,#1

PercentageLoop:

     mov  A,R0
     movc A, @A+dptr
     subb A, TL0
     jc   getPercentage
     inc  R0
     sjmp PercentageLoop

getPercentage:

     dec  R0
     mov  A,R0
     mov  B, #5
     mul  AB

;*************************************************************
; REGISTER A NOW HOLDS PERCENTAGE VALUE
;*************************************************************
Cap_Values:

     DB   238  ;0%  0
     DB   150  ;5   1
     DB   108  ;10  2
     DB   85   ;15  3
     DB   78   ;20  4
     DB   72   ;25  5
     DB   67   ;30  6
     DB   62   ;35  7
     DB   58   ;40  8
     DB   54   ;45  9
     DB   48   ;50  10
     DB   46   ;55  11
     DB   45   ;60  12
     DB   43   ;65  13
     DB   42   ;70  14
     DB   39   ;75  15
```

```
DB    37    ;80  16
DB    35    ;85  17
DB    33    ;90  18
DB    32    ;95  19
DB    31    ;100 20
```

## 8.2 Appendix B: Snapshot of the Automatic Soda Fountain Dispenser