UNIVERSITY OF NOTRE DAME-ELECTRICAL ENGINEERING

Low-Level Design

Team AutoBev

Elizabeth Clark, Lorena Garcia, Alex Macomber, Mark Pomerenke

2/18/2011

Contents

1 Introduction

In an environment where profit is dependent upon efficiency of product delivery, it is evident that a major source of missed revenue comes from lengthy wait times. A place where this problem is often seen is at overcrowded drinking establishments. Here, the drawn out process of ordering a drink, followed by waiting for completion of a transaction leads to much wasted time.
With the technology available in our current society, it is apparent that waiting times can and should be reduced. Much of the time associated with this process can be eliminated if the initial stage is automated. Through the utilization of a computer system that provides a means to order, pay for, and pour a beverage, without dependence on a restaurant employee, the time between request and delivery of a final product can be greatly reduced and streamlined.
The AutoBev system will be beneficial to any establishment that wishes to decrease wait times, increase the accuracy of orders, reduce the workload of employees and increase revenue.

2 Problem Statement and Proposed Solution

The efficiency of most drinking establishments today is less than spectacular. Patrons can spend the majority of their night waiting to be noticed by a bartender. Once noticed, the transaction between patron and server can last for several crucial minutes that could be spent on serving other customers.

In order for a transaction to be completed, a patron must first be recognized as the next waiting customer by the bartender. The patron then explains his or her order to the server who then proceeds to make the specified drink. After the drink is served to the customer, the bartender tells him or her the price of the order and the customer in turn pays the bartender by the preferred method of payment. Regardless of payment choice, the payment transaction is very time consuming. Bartenders must enter the order into a touch-screen, swipe a card or open the register, print a receipt and have the patron sign the receipt.

The entire process of ordering a drink at a bar can be both time consuming and a frustrating experience for the server and customer. If a bar were to be able to serve drinks more frequently, then its revenue would increase. Another problem with the current system is that patrons tend to be served out of order which in turn decreases customer satisfaction.
The proposed solution for a more efficient ordering and payment system at a bar is to have an interface that allows for customers to place their orders without assistance from an employee. Customers will swipe their credit cards to login into the system, order and pay for their drinks independently.

The customer has the choice to purchase a “pour your own” drink or to place an order with the bartender. If the customer chooses to pour their own, he or she will place a cup underneath an

adjacent tap and the press a button to dispense the beverage. Customers will have the option to purchase a preset amount (i.e. a 60 ounce pitcher) or pay per ounce.

3 System Description and Block Diagram

The AutoBev Dispensing System will increase the efficiency of an establishment by effectively restructuring the drink ordering process. The system will have an interface that allows the user to either pour their own beverage at the AutoBev Kiosk, or to select a drink to be added to a queue which will then be prepared by a bartender. The system will personalize the ordering experience by storing user information, preferences, and history into a database. This information will be displayed on the user interface during the ordering process. There will be communication between the user interface and the bartender interface. The bartender will receive drink orders and a corresponding drink order number from the user interface. The bartender will also be able to indicate to the user interface the order number that is being filled, and will be able to move through the drink queue after completing an order. The current order number being filled by the bartender will be displayed at all times on the user interface. The use of the system can be seen in Figure 1 below.

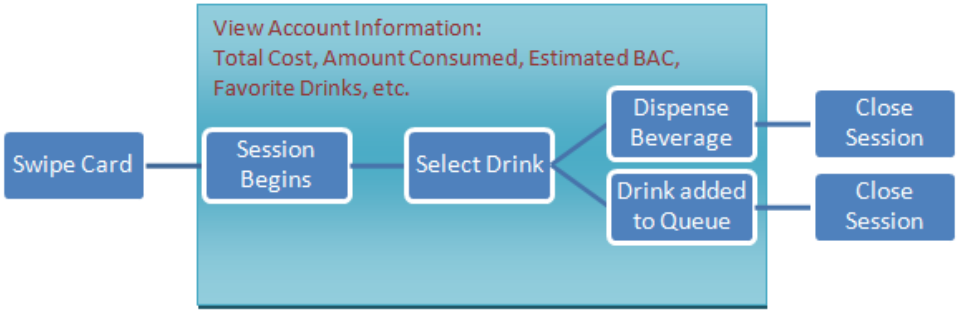

Figure1. System Use Flow Chart

The AutoBev system will require the use of a magnetic stripe reader at the front-end in order to read information from a user's credit card. This information will be used to create and access user accounts. After a card is swiped, a database will be queried to determine if the user has an existing account. If they do not already have an account, one will be created. A personalized session will then begin on the user interface and the user will navigate through the interface to select and order drinks.

The user interface will be in constant communication with the bartender interface so that the drink queue can be updated each time a new order is placed. It will also be in constant communication with the microcontroller so that the microcontroller will open and close the solenoid valve at appropriate times. Additionally, the microcontroller will communicate data from its sensors (flow and light) back to the user interface.

The database will store user account information and will be updated by the user interface upon completion of a session. The database will hold both static values of name and card

numbers, as well as dynamic values such as total volume consume, most popular drink, estimated BAC, and bill balance.

A block diagram of the entire system can be seen below in Figure 2.

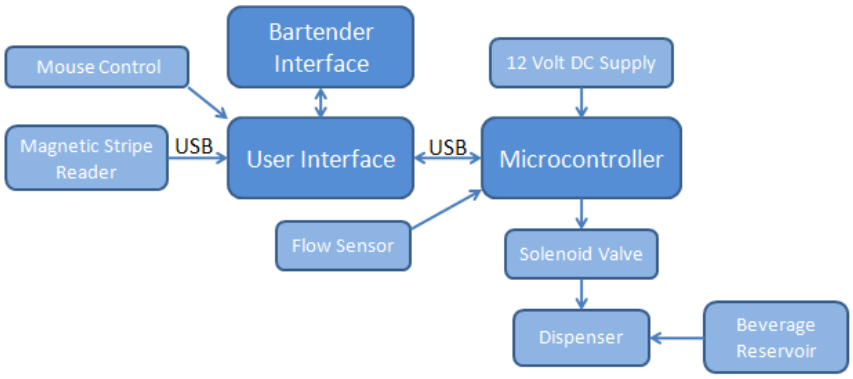

Figure 2. Block Diagram of System

4 System Requirements

4.1 Overall System:

Overall System Requirements	
General	Must be capable of receiving and processing drink orders digitally Must use simple user commands for navigation (single click) Must have a layer of protection between CPU and other sensitive electronics and users/beverages
Size	Must fit onto a floor area no larger than 2'x2' (not including any keg)
Power	Must be powered by wall plug in 120V AC
Compatibility	Must be able to connect to a standard keg Must be expandable to include more drinks Must be in English Must be in compliance with current health standards

4.2 Subsystem and Interface Requirements:

Magnetic Card Reader	
General	Must be able to scan credit cards and send information to the PC
Size	Must not be bigger than 6"x6"
Power	Must be powered through USB interface
PC Software	Must be able to interface to a PC through USB Must be able to emulate a USB Human Interface Device (HID) keyboard Must turn on and off with PC
User Interface	

General	Must be implemented on a standard monitor
Power	Screen must be powered by 120 V AC
Compatibility	Must work on Windows PC Will Be Programmed Using Microsoft Visual Studio Will Program in C# Visual
Bartender Interface	
PC Software	Must be capable of running on a low performance windows machine Must list drink items in order in a FIFO queue Must assign item number to new drink items for claims purpose (will not need to issue ticket) Must be capable of adding orders to the queue from remote client (customer interface) Must be capable of deleting queue items from local user Must be able to handle bi-directional communication via Local Area Ethernet Must be compatible with UI program Bartender Server must be multi-threaded for expandability
Microcontroller	
General	Must handle at least 8 digital inputs * Must handle at least 4 analog inputs* Must supply at least 4 digital outputs * Must indicate the state of digital outputs with LED's Must be programmable with C software Must have a universal asynchronous receiver transmitter (UART) Must be able to communicate serially with CPU over USB connection Must have non-volatile memory Must be capable of 3.3V operation. Must have On/Off switch Must have reset function *will only use 2/1/1 I/AI/O ports for demonstration, but built in for expandability
Power	Powered by 12 V supply from wall converter
Compatibility	Will receive signal from computer Will receive signal from flow sensor
Beverage Dispensing and Sensing	
General	Must have keg to pour drink from with appropriate keg components (spigot, keg adapter, pressure valve, tubing, and gas tank). Must have a solenoid valve to only allow beverage to pour through if already paid for. Must have a flow sensor to keep track of how much beverage has been poured. Must have Cup Senor to detect if cup is present Must have emergency stop
Power	Must be powered via plug into a 120 VAC outlet Must create intermediate regulated voltage power supplies of 12 V and 3.3V

Valve Permission Solenoids	Solenoids must receive 12 V across the coil to activate Must be able to control 12 V solenoid signal with a 3.3 V output Amplifier FET must be able to handle 0.53 A, and a max V_{ds} of at least 20V Must connect to 0.25” OD, .17” ID tubing
Flow Sensing	Requires specified resistive network for proper signal output Requires 5-24V power supply Must connect to 0.25” OD, .17” ID tubing
Cup Sensor	Must detect if cup is present underneath spigot Must ignore ambient light
Emergency Stop	Must allow user to stop beverage flow if anything goes wrong Should rely on hardware only (not software)
Miscellaneous	Tubing must be able to couple into 3/8” tubing
Microcontroller Software	Must use a reasonable amount of program memory Must be able to communicate with solenoid(open/close it).
	Must be able to get input from flow sensor periodically. Must be able to get input from cup sensor. Must be able to communicate with User
PC to Microcontroller USB Interface	
General	Must have USB interface User Interface PC must be able to communicate with microcontroller by sending bytes Microcontroller must be able to communicate with User Interface PC by sending bytes

4.3 Future Enhancement Requirements

Future Enhancement Requirements	
Online Database	System should be capable of uploading the data into an online database. This information will be accessed by the individual users who will have accounts registered with the website. Data will show drink purchase information such as time, quantity, type and expense.
Charge Credit Cards	Must be able to charge the credit card of the patron. This will be an upgrade from the current proposed version that simply stores the credit card information in a local database without charging the account.
Interfacing with Mobile Device	Must be able to use an smartphone application to complete beverage order and add the order to the queue Must Send text message to smartphone when drink is ready
Serve Mixed Drinks	Must be able to properly make and serve “mixed drinks”

Cognitive Testing	Must be able to determine if customer is in suitable state for another drink
Interconnect Multiple Kiosks	Must allow for multiple dispensing units to be interacting with a centralized bartender interface within a single establishment

5 Low Level Design

5.1 Card Scanner

5.2.1 Hardware Decisions

The magnetic stripe reader (MSR) will be placed alongside the customer interface so the customer can scan his or her debit/credit card. The scanner must be a “plug and play” device that simply connects to a USB port of a PC and emulates keyboard inputs without any additional software. The customer enters the graphical user interface (GUI) with the scan of his or her card. The system can then identify the customer and pull up his or her transaction history.

Unitech’s MS240 Magnetic Stripe Reader was chosen because the product meets all stated requirements and is relatively inexpensive. The MS240 interfaces with a Mac or PC and emulates a keyboard such that any running program will be unable to distinguish the MSR from any ordinary keyboard.

The benefit of the MS240 is that it is programmable. Using the Reader Configuration Manager, instructions can be downloaded to the MS240. These include but are not limited to: turning the beeper on and off, choosing which tracks to display, deciding how information is displayed and choosing the type of end character.

The card scanner will be programmed to output track one according to the ISO/IEC 7813 standard. This standard defines properties of financial transaction cards such as debit or credit cards. There are three tracks on the cards conforming to this standard. The format of track one which follows the format below.

Track 1, Format B:
Start sentinel — one character (generally '%')
Format code="B" — one character (alpha only)
Primary account number (PAN) — up to 19 characters. Usually, but not always, matches the credit card number printed on the front of the card.
Field Separator — one character (generally '^')
Name — two to 26 characters
Field Separator — one character (generally '^') **Expiration date** — four characters in the form YYMM. **Service code** — three characters
Discretionary data — may include Pin Verification Key Indicator (PVKI, 1 character), PIN Verification Value (PVV, 4 characters), Card Verification Value or Card Verification Code (CVV or CVK, 3 characters)
End sentinel — one character (generally '?')

Longitudinal redundancy check (LRC) — it is one character and a validity character calculated from other data on the track. Most reader devices do not return this value when the card is swiped to the presentation layer, and use it only to verify the input internally to the reader.
For example:
Track one of a Discover credit card:
%B0123456789012345^LAST/FIRST ^YYMMXXXXXXXXXXXXXXX?
Track one of a Visa debit card:
%B0123456789012345^LAST /FIRST M^YYMM XXXXXXXXXXXXXXXXXX?

5.1.2 Testing

In order to test the magnetic stripe reader, a word editor was opened and the MS240 was plugged into a USB com port. Ten unique cards were then swiped and the output track was verified. The cards varied between credit and debit and between card companies.

5.2 User Interface

5.2.1 Physical Unit

The user interface will be implemented on a standard PC monitor. The user will interact with the interface using a mouse. The choice to use a standard PC monitor, as opposed to a touch screen, was made after experiencing difficulties in finding a touch screen for a reasonable price. Regardless of the fact that the AutoBev system will be demonstrated using a standard PC monitor controlled with the use of a mouse, the system will also be compatible with a touch screen. This is because tapping a touch screen is interpreted in the same manner as is clicking a mouse. The AutoBev Kiosk will be built such that any standard computer can fit inside.

5.2.2 Software Design Decisions

In order to develop the user interface, we chose to create a GUI using the Microsoft Visual Studio IDE. After experimenting with a few of the languages supported by this environment, we decided that the best way to achieve a professional-looking and easy-to-use interface was to program using Visual C#. The combination of Microsoft Visual Studio with Visual C# has allowed us to access many useful predefined **libraries which** will help to enable communication between the user interface and both the microcontroller and the magnetic stripe reader**.**
Another useful feature of Microsoft Visual Studio is that it allows for the creation of a type of project called a “Windows Form Application.” Using this project framework, we have been able to create multiple forms, or screens, with which the user will interact. The ability to separate each step of the drink ordering process by creating a new form has greatly simplified the development of the interface.

5.2.3 Software Flow:

The user interface will be a means to display step-by-step instructions to the user in order to guide them through the process of both purchasing a specialty drink (i.e. one that needs to be made by a restaurant employee) and of pouring a beverage on their own.

Upon approaching the AutoBev System, the customer will be presented with the Welcome Screen Form. This will instruct the user to swipe their credit card. The information read from the credit card will be input into a text box located within the form (the MSR acts in exactly the same manner as a keyboard). This textbox will be continuously polled for changes in input. Anytime a new character is detected, a KeyPressed event is triggered that stores that character in a buffer. The last character in a credit card track is a question mark. Therefore, in order to determine when the credit card number has been completely read, each character detected by the KeyPress event will be compared to a question mark. When the question mark is detected, the characters in the buffer will be stored as the card number.

When a card scan is completed, a new instance of the class, 'Person' will be created. This class is located in the namespace, 'Customer.' This class has as attributes, 'cardNumber' and 'sessionTotal.' The'cardNumber' attribute will hold the current customer's card number for identification purposes. The 'sessionTotal' attribute will hold the amount that is to be added to the customer's current total purchase in the database. Each time a new customer uses the AutoBev dispensing unit, these attributes will be reset. A Person's 'sessionTotal' will always begin at $0.00.

Once the cardNumber attribute of the new Person has been set, it will be used to query a database to check for the existence of an account associated with that number. If an existing account is not found, one will be created. In addition to holding the card numbers of each customer that has used the AutoBev System, the database will also hold the user's total tab for the night. This information will be used for billing purposes when the user decides to close their tab.

The program will then progress to the Drink Type Selection Form as seen in figure 3. This form presents the user with two options. The user may chose to either order a drink that will need to be made by the bartender or to dispense their own beverage. The choice will dictate which form appears next.

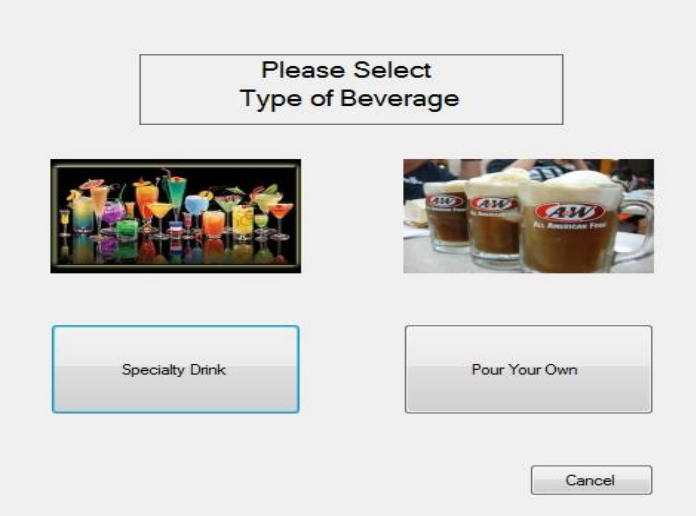

Figure 3. Type Selection Form

If the user chooses to order a specialty drink, they will be presented with a list of possible drink choices. Upon choosing a drink, the 'sessionTotal' attribute will be increased by the price of the drink (stored as a variable within the Specialty Drink Selection Form). Simultaneously, a function to send the drink order to the bartender interface will also be called in order to update the queue on the bartender interface. This drink order will be sent via a computer network.

If the user chooses to dispense their own beverage, they will be asked if they would like to pay by the ounce, or if they would like to dispense a predetermined amount of beverage. If they choose to dispense a predetermined amount, they will then be prompted with a screen that allows them to choose from a selection of sizes. Based on the selection of size, a variable called 'limit' will be set and passed to a screen that contains a start and stop button. If the user chooses to pay by the ounce, the limit will be automatically set to 221.6 oz (see Section 5.6.2 for explanation), and the user will be presented with the Start and Stop Pour Form, which can be seen below in figure 4.

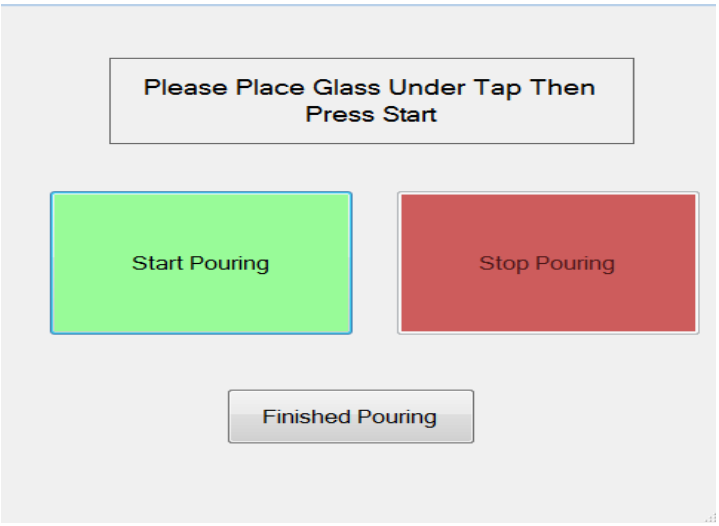

Figure 4. Start and Stop Pouring Form

As soon as the user clicks the start button on this form, a signal to open the solenoid valve will be sent to the microcontroller via USB. The bytes and the information that they contain can be

seen in section 5.6.2. The user may start and stop pouring multiple times within one order. The order will end when the user selects the finish option on the Start and Stop Form, or when the microcontroller sends a flag indicating that the limit of the order has been reached. At this time, the solenoid value will close and the AutoBev dispensing unit will stop pouring. Each time the microcontroller sends bytes to the interface, the amount of liquid dispense will be sent, regardless of if the user is paying per ounce, or dispensing a predetermined size. When the user chooses to finish ordering, that limit will be used to calculate the user's total charge. The sessionTotal attribute of the current instance of Person will be updated.

Upon finishing an order, the user will be prompted with the Thank You Form. This form will give the option of ordering again, or checking out. A user may choose to checkout by ending their current session, or by closing their tab for the night. In either case, the cardNumber attribute of the current person will be used to access their account within the database. The account total in the database will be increased by the charges stored in the sessionTotal attribute. The current instance of 'Person' will then be destroyed in preparation for a new session with a different user.

A user may order as many times as they please within a single session. The sessionTotal will be continuously updated during this time. Only at checkout will the database be queried for the 'cardNumber' and only at that time will the total purchase in the account be updated.

5.2.4 New Classes

Classes:

A new class called 'Person' will be created. It will exist in the namespace, 'Customer.' This class will have two private fields. The first is a private string called 'cardNumber.' This field can be accessed with the CardNumber property which contains set and get methods. These methods allow the program to update and access the private card number of the customer. The second private field is a private double called 'sessionTotal.' This field can be accessed with the SesssionTotal property, which also contains set and get methods. These methods allow the program to update and access the total charges that a customer has accrued within a single session. When the customer chooses to end a session, the field containing the card number
will be used to query the database and identify the customer. The amount contained in the session total field will be used to update the customer's total purchase for the night that is stored in the database.

5.2.5 Testing

The user interface will be tested by stepping through each path that a customer may take while ordering a drink. Through this process, it will be possible to verify the functionality of each command. In order to ensure that each function is executing when required or expected, verification statements will be output to a text file at strategic points within the interface. This text file can then be opened and will essentially display the progression of the customer through the interface.

Additionally, each button on every form will be tested. This can be done deterministically by pressing every button on every form and verifying that the appropriate next form appears.

Various values will be input for card numbers to make sure that the program operates in the expected manner, and is capable of determining when an input does not make sense.

Once the interface is integrated with the microcontroller and flow sensing hardware, tests will be run to ensure that the volume returned by the microcontroller is in fact the volume that has been dispensed. It is vital that this measurement is extremely accurate for charging purposes.

In order to make the software as resistant to errors as possible, 'try-catch' statements will be added within each function. These statements will guarantee that any error that is encountered during program execution will be handled in a systematic manner. It will also ensure that an error encountered during run-time will not cause the system to crash.

Although it will not be possible to write computer-testing algorithms for the user interface, by writing code with error-handling capability and through careful testing, it will be possible to ensure the functionality of the user interface. The possible paths that a user can take through the interface can be seen below in figure 5.

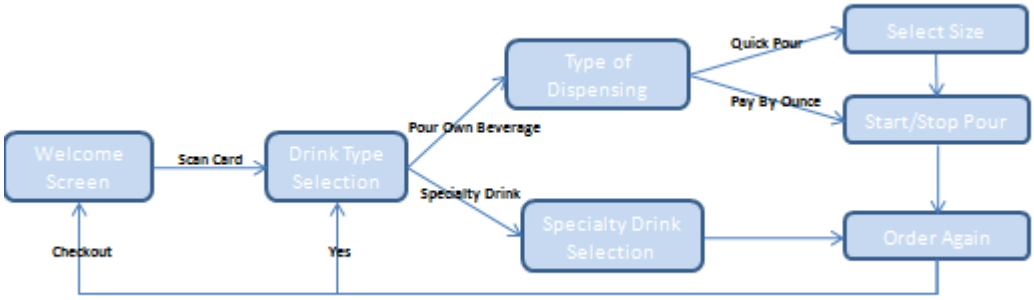

Figure 5. Flow Chart Demonstrating Paths Through User Interface

5.3 Bartender Interface

5.3.1 Interface Design

The bartender server will serve as a database of recently ordered specialty drinks. The software must allow the bartender to read drink entries made by the customer at the user hub, and then delete the orders after they have been prepared. This must be a fairly simple display in order to speed up drink production, and will only entail a drink list updated with additions from the User hub, and options for the bartender to scroll through the orders and delete selected drinks.

A UDP Server/Client protocol will be used for communication between the two computers, the User and Bartender hub, over a local area connection using Cat 5e Ethernet Cable. A UDP protocol was chosen because the amount of information being communicated will be extremely small and will only need to be sent occasionally in packets, ideal for UDP. The protocol must be written in C# so that it will seamlessly interface with the customer interface. The client UDP protocol program will be embedded within the UI program as a separate function file, which will be called by the UI program when a specialty drink is ordered.

The bartender interface will act as the server with capability to open multiple communication threads. This will allow the owner to add multiple machines to the system both locally and remotely by simply connecting the system to the internet.

5.3.2 Testing

The bartender interface will be tested by verifying that the drink selected on the user interface is properly displayed on the bartender interface along with its assigned drink order number. This process will be repeated for each possible drink selection. As drinks are ordered, it will be necessary to test that the bartender interface is updating correctly. Due to timing constraints, it will not be possible to write code to implement these tests. Therefore, the testing will be carried deterministically. Deterministic testing will also be used to ensure that the bartender can update the drink order that is currently being filled that is displayed on the user interface.

A flow chart of the UDP Client/Server with functions can be seen in figure 6.

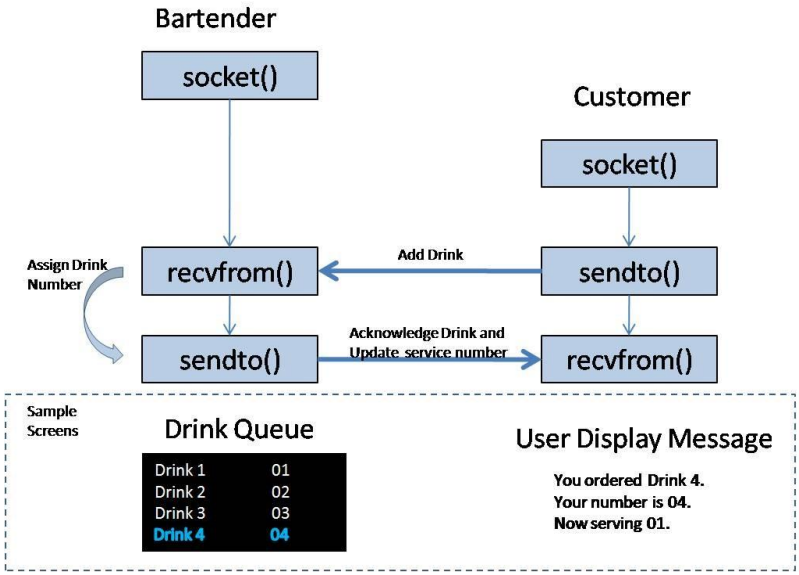

Figure 6. Flow of Bartender Interface

5.4 Microcontroller

The microcontroller used as the embedded intelligence for this project is the PIC18LF6722, which meets all of the requirements specified in section 4.2. A schematic of this microcontroller with the pins connected to I/O signals can be seen in the figure 7 below. Further information on the microcontroller function and additional circuitry and hardware that must be implemented on the board the microcontroller will come on are discussed in the following sections.

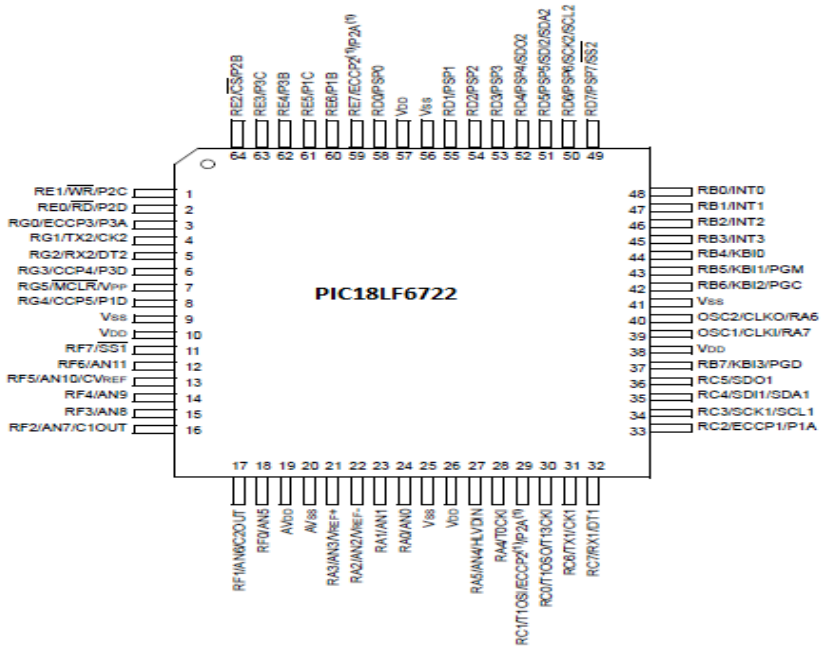

Figure 7. Microcontroller Schematic

5.5 Beverage Dispensing and Sensing

After a drink has been ordered, this feature will allow for drinks that do not need to be made by a bartender to be instantly available. The computer that serves as the user interface will communicate with the microcontroller via a USB interface, which will control all functions associated with a keg. Customers should be able to place their cup under a tap and manually pour their selected beverage once they have completed all necessary steps on the user interface. The mechanical system to control flow will be created to connect to a keg of liquid. In order to keep a constant pressure in the keg, a gas tank with a constant pressure valve will be needed. One of the team members already has a kegerator and the rigging. The tubing for the keg has been purchased separately. Thus, the needed spigot, keg adapter, pressure valve, gas tank, and tubing are already available. The solenoid valve will be attached to the part of the tube close to the tap and will be actuated by a digital output from the microcontroller. The microcontroller will signal the solenoid to open if a customer has placed their order and is manually trying to pour their drink. A flow sensor will be placed in the flow path to send a signal to the microcontroller which will specify how much drink has been poured for pricing purposes and to monitor flow in case the customer opted to have a specific amount of drink poured rather than to pay by ounce. We will also have a cup sensor by the tap to indicate that a cup has been placed there in the chance that someone is trying to pour a drink without a cup. Additionally, we want to utilize an emergency stop that is independent of the software in case anything goes wrong and we want to stop beverage flow. Upon completion, the microcontroller should send information back to the user interface so that the tab can be updated.

Note that our system is designed with two flow sensors, two cup sensors, and two solenoids in order to handle multiple beverages. The microcontroller board will be designed to hand up to four of each to allow for expansion, so that a venue may offer

more beverage options. The actual system demonstration will only utilize one of the sets which will be attached to the output of a Danby 5.8 Cu. Ft. Capacity Keg Cooler (Model # DKC645BLS).

5.5.1 Flow Sensing

The SwissFlow SF800 flow sensor will be supplied by a 12V source and will require a resistive network as seen in figure 8 below. The output of the sensor is a pulsed output that generates 5600 pulses/liter. The flow sensor will allow the microcontroller to know how much liquid has been poured. By counting the pulses we can calculate the volume and flow rate of the liquid that is passing through the tube.

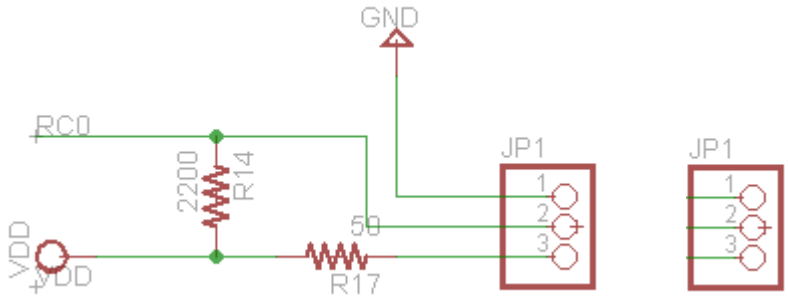

Figure 8. Flow Sensor Circuit

A cup sensor will act as a safety for the system such that a beverage will not be dispensed unless there is a cup present beneath the spigot. An OSRAM 30 mm Proximity Sensor (SFH 7741-Z) will be used to detect the cup and will communicate the cups presence to the microcontroller via an analog input. The sensor will require a 2.4-3.6V power supply which is satisfied by the microcontroller 3.3 V source.

5.5.3 Valve Permission Solenoid

The solenoid serves as the main action unit as it will open or close to allow or stop the beverage from flowing. The ECT 12 V solenoid (US5311162) is a normally closed valve that will open when 12 V is applied to the coil. The internal resistance of the solenoid coil is about 23 ohms, therefore to switch open the solenoid requires about .53 amps. A power MOSFET, NDS355, was chosen appropriately for the coil driving circuit, which is driven by a 3.3V digital output applied to the gate, this digital output will have a corresponding indicator LED for troubleshooting so that the owner can check the status of the solenoid. The driving circuit can be seen in figure 9 below.

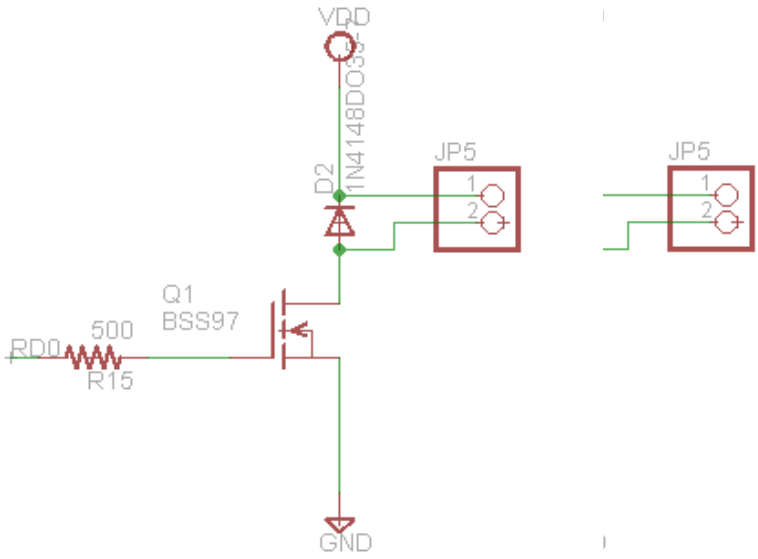

Figure 9. Solenoid Circuit

5.5.4 Emergency Stop

As a final failsafe the circuitry will also involve an emergency stop button that will close the solenoid in case of error. This will provide an easy way to manually close the solenoid, and will be implemented as a toggle switch that will connect or disconnect the solenoid from VDD.

5.5.5 Microcontroller Software

The control of the beverage dispensing and sensing subsystem relies on a program written in C that is programmed into the microcontroller. The main aspects of this program are discussed in the following paragraphs.

This program will rely heavily on the use of several interrupts on Port B of the microcontroller for flow control. One interrupt flag comes from bit 1 of Port B and is triggered by pulses from the flow sensor. The other interrupt flag comes from bit 2 of Port B and is triggered by pushing a button to indicate the desire to pour a beverage. This interrupt is solely for testing purposed to show functionality of the drink dispensing subsystem. For the completed project the button and corresponding interrupt will be removed as the desire to pour will now be sent from the user interface to the microcontroller via USB. Another interrupt will also be on PortB and will correspond to the cup sensor. This interrupt has not yet been established in our working code. There are three current variables that will be effected by these interrupts. These are *increaseVolume*, *volFlag*, and *openSolenoid*. Each variable is initially set to false and will be changed to true at different points in the program. The first variable, *increaseVolume* will be true when an interrupt occurs indicating a pulse from the flow sensor is detected. The variable *openSolenoid* will be true when an interrupt occurs indicating the button is pressed. The *volFlag* variable will be true when a volume limit set later in the program is met. It is necessary that the interrupt functions come before the main program so that they

will work properly. When the two given interrupts are sensed they will change the state of the previously mentioned variables and reset their given interrupt flags, so as to escape the interrupt. Our program must also have a variable for the cup sensor input, which will be set to true when a cup is sensed and to false when no cup is sensed. This variable should also be initialized to false before the main program.

In the main program it is necessary to set intcon, intcon1, and intcon2 to the appropriate high and low bit sequences so as to enable global interrupts, intedge interrupts, and the interrupt flag. The LCD screen is to be initialized for testing purposes. Input and output ports will be set on Port B. These will include inputs for the flow sensor pulses and button, so the microcontroller can receive signals to monitor the flow. An output will also be set for the microcontroller to be able to send signals to the solenoid in order to have it close or open as desired. The variable *volume* will be initialized to 0 for use in the main function. An appropriate message will print to the LCD and terminal for testing purposes to indicate that the button be pushed for beverage pouring to begin. At this point the main function will have a while loop that does nothing while waiting for the button to be pressed and for a cup to be sensed. When both of these given variables are set to true the program will escape the while loop and move on.

When the button is pressed *volFlag* will be set to false and a high output will be sent to the solenoid in order to open the valve and a message will be printed for testing purposes indicating that the beverage is being poured. The variables *conv*, *inc,* and *limit* will be set to 165.6, 1, and 1400 respectively. The *conv* variable indicates the amount of pulses/ounce given by the sensor. The *inc* variable is to allow for increments of 1 for each pulse. The *limit* variable is to allow us to check that our subsystem works in dispensing a specific amount of volume and is set to 1400 for testing purposes. The program will enter a while loop that will continue looping until the *volFlag* variable is true, or when the limit of 1400 pulses is reached. Within the while loop, if pulses from the sensor are received the *volume* will increment by *inc* and the interrupt flag for the pulses will be reset. While this is happening, if *volume* exceeds *limit* a low signal will be sent to the solenoid indicating that it should close and *volFlag* will be set to true, which will cause the program to exit the while loop. The amount of beverage poured will be calculated via the equation (*volume-inc)/conv* and this value will be printed to the terminal and LCD. This will aid in the testing of the accuracy of the beverage dispensing subsystem as we can compare the measured volume to volume that we measure physically before testing.

We have already written a rough draft of the program to be used based on the functions and variables explained in the above paragraphs and flow chart. We have been programming updated versions of this program to the microcontroller provided to us in the project kit via the USB also provided to us. Testing has been done by connecting the appropriate input and outputs to Port B as indicated above and in the following code. For testing of the subsystem we have been filling a bottle with water and allowing the water to flow through the tube that is connected to the solenoid. This program has been executing successfully for testing and will continue to be used and modified as needed. We currently need to modify it to include the cup sensing variable. The basic logic behind this has already been explained above and is currently noted through comments in the code. The current code can be seen in figure 10 below.

----------/*

```
AutoBev
Drink Dispensing and Control Program
*/

#include <system.h>
#include "task3lib.h"
#include "LCDlib_4.h"
#pragma DATA _CONFIG1H, _OSC_HS_1H
#pragma DATA _CONFIG2H, _WDT_OFF_2H
#pragma DATA _CONFIG4L, _LVP_OFF_4L & _XINST_OFF_4L
#pragma DATA _CONFIG3H, _MCLRE_ON_3H
#pragma CLOCK_FREQ 20000000

// variable for INTCON register interrupt flag volatile bit toint@INTCON.2;
volatile bit intedge1@INTCON2.5;
volatile bit int1IF@INTCON3.0; // interrupt flag for port b, bit 1 (flow sensor pulses)
volatile bit int2IF@INTCON3.1; // interrupt flag for port b, bit 2 (input from button)
//set up interrupt for cup sensor

bool increaseVolume = false; // true when interrupt occurs indicating flow pulse detected
bool volFlag=false; // flag indicating that the volume limit hasn't been reached
bool openSolenoid = false; // flag indicating solenoid should be closed, true when button
pressed
//bool cupSensed=false;

void interrupt(void){
if(int1IF){
increaseVolume=true; //Indicate that there is something to do int1IF=0; //reset
}
if(int2IF){
openSolenoid=true; // indicate button has been pressed and allow liquid to flow int2IF=0;
//reset flag
}
}
//add interrupt for cup sensor void main(void){
intcon=0b10000000; // enable global interrupts intcon2=0b00110000; // enable intedge
interrupt intcon3=0b01011000; // enable interrupt flag
LCD_init(1);  // initialize LCDs trisa.1=0;
trisa=1;
trisb.1=1; // set port b.1 to input (pulse input from flow sensor)
trisb.2=1; // set port b.2 to input (input from button)
trisb.6=0; // make this an output port, to control solenoid, set high to open solenoid valve
// add an input to detect highs/lows from cup sensor

unsigned short volume = 0;

init_usart(129); //Set baud rate

hypTerm_printStr("Welcome to AutoBev...  Press the red button to start pouring...  ");
LCD_printf("Welcome to AutoBev...  Press the red button to start pouring...  ");
```

```
while (openSolenoid==false){ //add cupSensed=false to this statement
} //do nothing while waiting for button press

latb.6=1; //send high signal to solenoid to open valve hypTerm_printStr("Pouring your
beverage...   ");

float conv = 165.61176576; //Flow sensor: ~165.61176576 pulses/ounce
int inc = 1;     //set increment value (volume increases by 1 unit each limit pulses)
int limit = 1400;         // set volume limit (# of pulses before increment)

//while loop runs until the volume limit has been reached while (volFlag==false) {
//when a pulse is received from the flow sensor, this if statement executes if
(increaseVolume) {
volume=volume + inc; // volume in milliters increaseVolume=false; // reset flag
// executes if the volume is greater than the limit to close solenoid valve if (volume > limit)
{
latb.6=0; // turn off solenoid, close the value volFlag=true; // break out of while loop
hypTerm_printStr("Finished Pouring...  Amount Poured: ");
hypTerm_dec((volume-inc)/conv); //display the amount of volume poured
hypTerm_printStr("    \n");
} // close volume if loop
} //close increaseVolume if loop

} //close infinite while loop

}
```

Figure 10. Autobev Dispensing Control Program

5.6 PC to Microcontroller USB Interface

The computer interface is going to communicate with the microcontroller in order for the software program controlling the drink dispensing system to know whether or not the user wants to pour a specific amount, pour an unknown amount of beverage, or if a drink has even been
ordered.

5.6.1 USB Hardware:

To convert the asynchronous serial messages transmitted by the microcontroller into the standard USB signals transmitted by the PC through the COM port, an intermediary is needed. The device we have chosen to do this is the FT232Rl from FTDI. Professor Mike Schafer of the University of Notre Dame designed the circuit governing the operation of this device. The circuit is used with his permission and shown in the following figure:

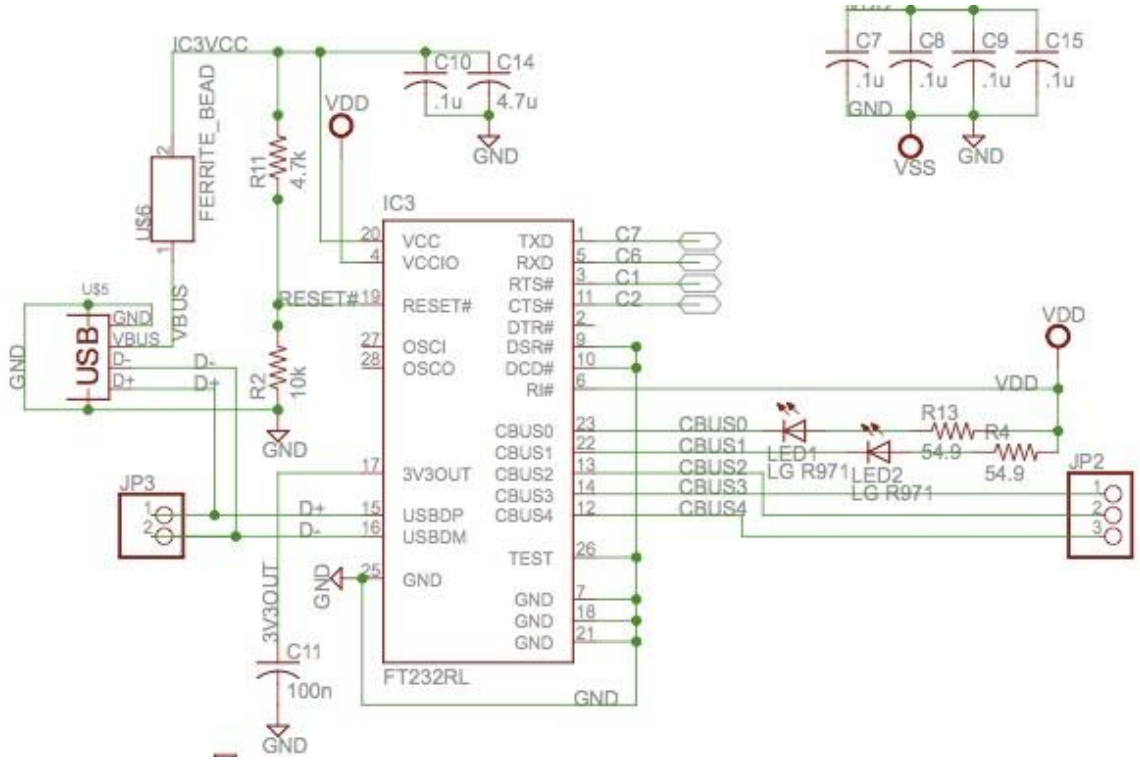

Figure 11. FT232RL Schematic

This USB signal from the PC carries a 5V signal. This 5V signal will be used to power the board. This 5V voltage is run directly into a 3.3V regulator, as shown in the following figure:

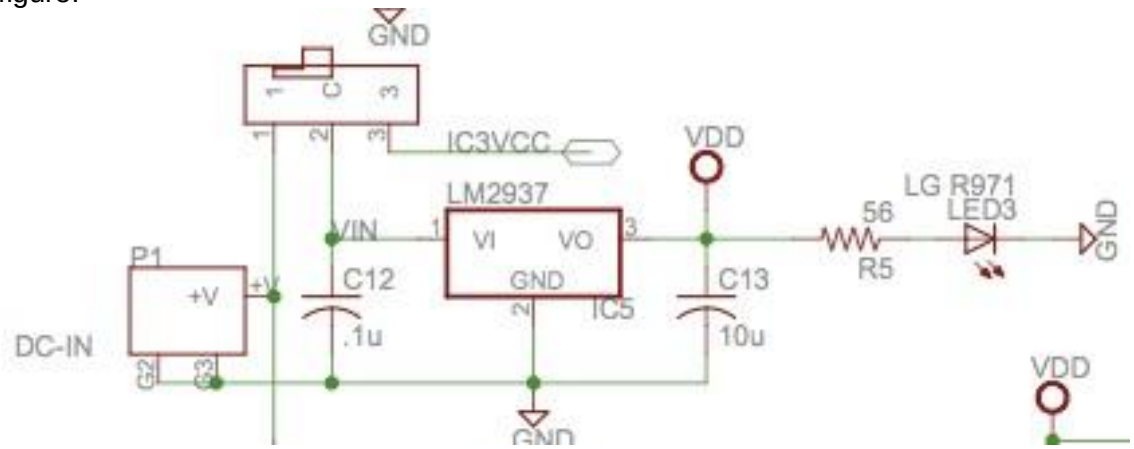

Figure 12. Regulator Circuit.

5.6.2 Testing:

To test this USB protocol, a variety of bytes will be sent from the microcontroller to a terminal. If all these bytes can be received correctly, than microcontroller transmission and PC reception have been verified. Then, a variety of bytes will be sent from the PC program to a microcontroller and displayed on an LCD screen. If these bytes are received correctly, then microcontroller reception and PC transmission will have been verified. Finally, the microcontroller and the PC program will be connected together for two-way

communication. If successful, the USB protocol will pass this test. The following section goes more into detail of the protocol we plan to have working for the final project.

5.6.3 PC to Microcontroller Protocol

There are two modes of communication, one in which a drink has not been ordered and one in which a drink is not being poured. These modes are summarized below.

Mode 1: A Drink Has Not Been Ordered

In this mode the microcontroller will sit idle waiting to hear from the PC. When a customer decides to pour his or her own beverage, a single byte will be sent to the microcontroller to indicate either the size of the drink order, or if the customer is paying per ounce. The microcontroller responds to the PC during the next mode, i.e. after it has started pouring.

Mode 2: A Drink is Being Poured

Once the microcontroller starts pouring, it will send three bytes to the PC every time it has poured 0.091 ounces (corresponding to 15 pulses from the flow sensor). These three bytes will contain the status of the microcontroller (if it is still pouring and if a cup is there) as well as a 16-bit number that contains the amount of volume that has been poured. The PC will interpret the value received in units of 100µL. For example, if the microcontroller sends a value of 5, the PC knows that 500µL have been poured. Using this size unit will enable a maximum value of 221.6 ounces (or 3.6 pitchers) to be sent over the interface with a resolution of 0.00338 ounces.

When the PC receives these three bytes from the microcontroller it will respond with the same type of byte that it sent in the first mode. This time however the message acts as an acknowledgement that the microcontroller is pouring and that it should continue.

PC→ Microcontroller

Byte 1:

Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Success/Error	Message Type	Start or Stop	Order Type	Size 1	Size 2	Size 3	Filler
1 = success 0 = error (resend)	1 = start/stop 0 = new order	1 = start pouring 0 = stop pouring	1 = preset size 0 = pay per oz.	1 = 12 ounce 0 = no order	1 = 16 ounce 0 = no order	1 = 60 ounce 0 = no order	DC

Microcontroller → PC

Byte 1 (Status):

Bit 0	Bit 1	Bit 2	Bit 3-7
Success/Error	Status	Cup Sensor	Filler
1 = success 0 = error (resend)	1 = still pouring 0 = done pouring	1 = cup present 0 = no cup	DC

Byte 2 (First 8 bits of 16-bit value of volume):

Bit 0-7
Bits 0-7 of Volume Poured Value (in 100µL)

Byte 3 (Last 8 bits of 16-bit value of volume):

Bit 0-7
Bits 8-15 of Volume Poured Value (in 100µL)

6 Bill of Materials

Subsystem	Description	Distributor	Part #	Quantity	Unit Price	Total Cost
FS&CC	SwissFlow SF800	Swissflow		2	0.00	$0.00
FS&CC	ETI 12V Solenoid (US5311162)	stock		2	0.00	$0.00
FS&CC	MOSFET N-CH 30V 1.7A SSOT3	Digi-key	NDS355ANCT-ND	2	0.51	$1.02
FS&CC	50 ohm Resistor	stock		2	0.00	$0.00
FS&CC	2200 ohm Resistor	Digi-key	RHM10.0KCRCT	2	0.00	$0.00
FS&CC	500 ohm Resistor	stock		2	0.00	$0.00
FS&CC	Diode 1N4001	Digi-key	1N4001FSCT-ND	2	0.31	$0.62
FS&CC	toggle switch	stock		2	0.00	$0.00
FS&CC	Ander-Lign Compression Connector (Brass)	Home Depot		2	2.19	$4.38
FS&CC	Clear Vinyl Tubing	Home Depot		1	2.98	$2.98
FS&CC	Danby 5.8 Cu. Ft. Capacity Keg Cooler	loaned		1	0.00	$0.00
FS&CC / Micro	12v voltage reg.	Digi-key	497-1210-1-ND	1	0.81	$0.81
Card Scan	Unitech America MS240 Mag Stripe Reader, MSR Track I, II&III, USB	Unitech America (via provantage.com)	Manufacturer Part# MS240-1T2	1	37.18	$37.18
MicroController	slide DPDT switch	Digi-key	SW116-ND	1	0.81	$0.81
MicroController	300 ohm SMD resistor	Digi-key	P300DACT-ND	10	0.20	$2.04
MicroController	10k SMD resistor	Digi-key	P10KDACT-ND	2	0.00	$0.00
MicroController	4.7k smd resistor	Digi-key	P4.7KDACT-ND	1	0.00	$0.00
MicroController	.1uF smd capacitor	Digi-key	ECJ-2VB1E104K	3	0.00	$0.00
MicroController	10uF capacitor	Digi-key	PCC2225CT-ND	1	0.00	$0.00
MicroController	4.7uF capacitor	Digi-key	PCC1842CT-ND	1	0.00	$0.00
MicroController	red led	Digi-key	160-1422-1-ND	1	0.00	$0.00

MicroController	green led	Digi-key	160-1423-1-ND	9	0.00	$0.00
MicroController	diode	Digi-key	1N4148WTPMSCT-ND	1	0.00	$0.00
MicroController	ferrite beed	Digi-key	445-2201-1-ND	1	0.00	$0.00
MicroController	3.3v voltage reg.	Digi-key	497-1235-1-ND	1	0.00	$0.00
MicroController	20MHz ceramic resonator	Digi-key	490-4717-1-ND	1	0.00	$0.00
MicroController	USB connector	Digi-key	609-3656-ND	1	0.00	$0.00
MicroController	FT232RL-REEL	Digi-key	768-1007-1-ND	1	0.00	$0.00
MicroController	PIC18LF6722-I Microcontroller	Digi-key	PIC18LF6722-I/PT-ND	1	0.00	$0.00
MicroController	Power Connector	Digi-key	CP-002A	1	0.00	$0.00
MicroController	Board Programmer	Provided in kit		1	0.00	$0.00
Bartender Interface	25' Cat5e Ethernet Cable	Provided		1	0.00	$0.00
Bartender	LC Computer	Provided		1	0.00	$0.00
User Interface	LC Computer	Provided		1	0.00	$0.00

Figure 13. Bill of Materials

7 Conclusions

This project addresses a problem that limits the revenue of service establishments such as bars and restaurants. The proposed solution will eliminate the need for a server to deal with the time consuming payment process. We acknowledge that throughout the completion of this project, many obstacles will be encountered. The implementation of software subsystems that must communicate with one another will require extensive research and a good deal of learning. The hardware portion is also an extensive portion of the project that will take time and consideration.

If our group succeeds, we expect that our project will help to increase the efficiency of drinking establishments through the automation of the ordering and payment process. The AutoBev system is a marketable design that can be easily integrated into any service establishment.

8 References

8.1 Spec Sheets

Flow Sensor: http://www.swissflow.com/en/SF800/applications/food_and_beverage

Microcontroller: http://parts.digikey.com/1/parts/563243-ic-pic-mcu-flash-64kx16-64tqfp-pic18lf6722-i-pt.html

8.2 General Information

Magnetic Stripe Card: http://en.wikipedia.org/wiki/Magnetic_Stripe_Reader

Keg Cooler: http://www.coolerdirect.com/keg-coolers-danby-dkc645bls-3370-prd1.htm