4-16-2010

Final Design Report of the Automated Beverage Dispenser

Ryan Sollars
Trinity University

Iuri Gagnidze
Trinity University

Dylan Nealous
Trinity University

Chad Oian
Trinity University

TRINITY UNIVERSITY

Final Design Report of the Automated Beverage Dispenser

ENGR-4381

4/16/2010

Ryan Sollars, Iuri Gagnidze, Dylan Nealous, Chad Oian

Dr. Peter Kelly-Zion

At public events and festivals, a beer vendor's primary problem is that they are unable to serve customers quickly enough to meet the excess of demand. With so many people requesting service and so few serving, waiting in long lines has become commonplace at festivals and events. These long lines slow down business, which deters additional customer sales, resulting in a loss of profit for the vendor. This report discusses a solution to this problem. It is an automated beverage dispenser. It takes orders from a user and then pours out the specified drinks without human assistance. The removal of a person from the actual task of pouring a beer allows the vendor to take money and check identification of the customer while the machine pours their order. Having these actions performed in parallel optimizes the overall process of serving customers quickly. The machine is intended to increase the total output of a single vendor, resulting in increased profits and happier customers.

Table of Contents

Table of Figures

Table of Equations

Cxel Summi

1. Introduction

Long lines are a source of frustration for customers at festivals and public events. In order to purchase a beer the customer must go to a special vending booth to make his purchase at which point the vendor must perform a series of tasks. He must: a) take the customer's order, b) verify his age, c) pour the beverage and d) hand it to the customer, whereupon, the vendor repeats the process for all the customers in line. Alternatively, the servers could dedicate a single person to each task in order to increase speed, but this also increases the wages the vendor must pay to employees and could reduce the overall speed of the process due to limited working space. Currently all vendors' serving methods generate long lines because service speed cannot match demand, and when vendors cannot meet customer demands people become frustrated. This can lead to lost profit, as many people do not want to wait in long lines. An increase in the serving speed of each vendor would alleviate frustration for the attendees, making them happier and more likely to purchase beer.

The use of the *Automated Beverage Dispenser* would increase the vendors output while only utilizing minimal staff. Each vending station at a typical event or festival is assigned a lot with an approximate area of forty square feet [1]. The vendor must fit all equipment and resources into this area, with enough space left for all operations. The design is small and space efficient. It needs approximately six square feet of table top space. This is advantageous to vendors because fits the operation of pouring a beer into a smaller space than if the vendor dispensed the drink himself.

This project specifically addresses the problem of serving speed for public events and festivals. It must also maintain standards for accuracy and quality as well as space efficiency.

1.1. Design Goals

The primary concern of this project is to produce an Automated Beverage Dispensing Machine, capable of serving beer. In our design we have taken into consideration certain requirements. ~~In our design we have taken into consideration certain requirements.~~ Unlike other types of beverage dispensers, beer serving machines require industry rated tubing to transport beer due to certain health concerns. Also, the design must incorporate a nozzle that

1. San Antonio Conservation Society

reduces foaming during pouring. The design should also meet these general criteria to be considered a feasible solution to the design problem.

- Serving accuracy - The serving accuracy of the design consists of pouring a beer within plus or minus half a fluid ounce of a desired volume.
- Speed of service - The speed constraint gives a time limit of one minute to the serving process. The serving time begins when an order is received and ends when the final beer is full.
- Political and Legal - The design should meet all federal regulations for alcohol and beer sales. The task of handling payment and verification of customer identification is handled by the user. This removes liability from the machine and ensures that the operator is liable for selling beer lawfully.
- Economic - The project is limited to a budget of $1200. However, it is important to make decisions from an economic standpoint in order to reduce cost to the end user of the machine.
- Health and Safety - The health and safety of the customers and vendors are paramount to the engineering code of ethics. The machine makes use of electricity and pressurized fluid lines. In order to ensure operator safety the machine must not allow these two to mix. Customer safety is addressed by the use of sanitary tubing for fluid lines the machine uses, in an effort to prevent particulates in the beer.
- Size - The machine is required to fit on a long table [2]. It will rest comfortably on a tabletop surface.
- Durability - This automated machine is capable of functioning outside during weather conditions typically seen at festivals. Potential conditions can vary from a Hot sunny days to a humid day with mild rain
- Mobility - Set up and transportation of the equipment requires little effort. The design is mobile enough to be moved by two people.
- Convenience and ease of maintenance - The interface the operator uses is designed to be convenient and ergonomic. Maintenance for the machine as a whole can be

2. Long Tables are 2' x 6'

completely by a single person without risk of damage to the design. When a part fails it can be easily replaced.

- Repeatability - Finally, the repeatability constraint consists of the design's ability to pour multiple successful cups in succession. This simulates the working environment at a festival, and the machine needs to perform perfectly to be considered a good solution to the design problem.

During the designing process the project objectives changed to reflect a revision in the project's scope or direction. The project aim changed from a standalone machine to a vendor operated automated dispenser. The three revisions of the original project objectives include: removing the constraint to restock or refill the machine in less than 5 minutes, removing of a method of age verification to prevent the sale of alcohol to minors, and changing the measurement of pouring accuracy from one percent of total volume to half of a fluid ounce. Originally, a self contained and fully automated system was thought to be an appropriate solution for the design problem. This idea was deemed infeasible because it was out of the scope of the budget. It required a worker to physically replace the keg the system utilized. However for typical festival activity this is too strenuous for a single individual. It was decided that the objective should changed. Age verification was removed because it was out of the scope of the project. Texas state law does not allow for the sale of alcohol by a stand-alone machine. A human must be present in order to serve alcohol. For this reason, the age verification system was dropped. This eliminated the need for some programming and debugging in order to meet this objective. The accuracy constraint was changed from a percentage to a fixed volume because the user requires a fixed volume poured regardless of cup size. The percentage constraint, allows for greater error in larger volumes, and requires the machine to pour much closer to a set point for smaller volumes. For these reasons, the fixed volume error is the working constraint for the project. This reduction in scope makes the design process less complex.

Separate Section

2. Design Description

The final product consists of three subsystems controlled by a single microcontroller and various electronics. These subsystems include the Cup Dispenser, the Turntable and the Pouring

System. In addition to controlling these subsystems, the microcontroller is also handles the user interface.

2.1. Cup Dispenser

The Cup Dispenser must solve the problem of dropping a single cup into a hole in the cup plate. The Cup Dispenser has multiple components which can be seen in Figure 1: Cup Dispenser Components, and on the Materials list of Appendix B in more detail. Parts of the subsystem include:

- 1 – cup cylinder (3 ft tall)
- 2 – cup-screws (custom built seen in Figure 2)
- 2 – dc motors
- 1 – U-shaped fitting brace (Seen in Figure 1)
- 1 – metal stand (made by welding flat stock, square stock and a right angle brace, also in Figure 1)
- 4 – bolts and nuts for fitting brace to dc motor connection
- 1 – bolt and wing nut for metal stand
- 2 – bolts and nuts for metal stand

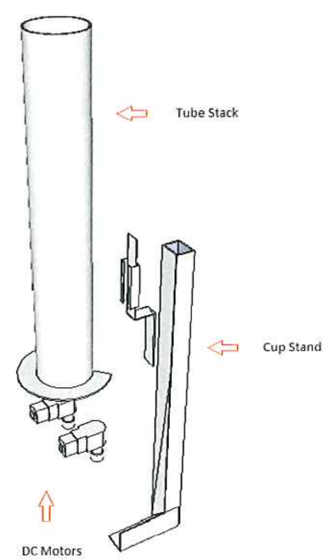


Figure 1: Cup Dispenser Components

The first action for filling an order begins with the Cup Dispenser subsystem. In order for a cup to be dropped into the Turntable, the microcontroller must first verify that there is no cup placed in the slot positioned under the Cup Dispenser. This is done with an infrared sensor and will be described in more detail in the electronics section. At this point two DC motors, controlled by the microcontroller, are activated simultaneously. These motors are connected to two custom built cup screws as seen in Figure 2: Cup Dispenser Motor with Cup Screw.

Figure 2: Cup Dispenser Motor with Cup Screw

The cup screws rotate in a motion that separates a cup from the stack. This separation process is done by feeding the lip of the cup into the grooves of the cup screw shown in Figure 3: Cup Screws engaged with Lip of Cup. The grooves in the picture are highlighted by black lines.

Figure 3: Cup Screws engaged with Lip of Cup

The cup screws rotate until a single cup is dropped. This process is repeated for every cup dropped. After a cup is dropped it is transported to the pouring system by the Turntable.

2.2. Turntable

The Turntable Subsystem consists of a base piece, a Lazy Susan gear, and a cup plate. The base piece has four legs, made from flat stock steel flanged at one end with a 1/8" hole in the flange. At the bottom of each flange an inch long piece of square stock is welded to elevate the whole Turntable. The other end of the leg is welded to a hexagonal piece of metal with a one inch diameter hole in its center. The legs and hexagonal piece make up the base piece of the Turntable. The Lazy Susan gear is a piece with two plates three inches by three inches connected together by a circular groove with ball bearings inside. This groove allows the gear to rotate. The cup plate is an aluminum disc that has six, three inch diameter holes spaced evenly around the center of the circular plate. All of these pieces can be seen in Figure 4: Turntable Subsystem Parts. The motor that provides the torque necessary to turn the cup plate can be seen in Figure 16, and the gearing on the motor can be seen in Figure 5.

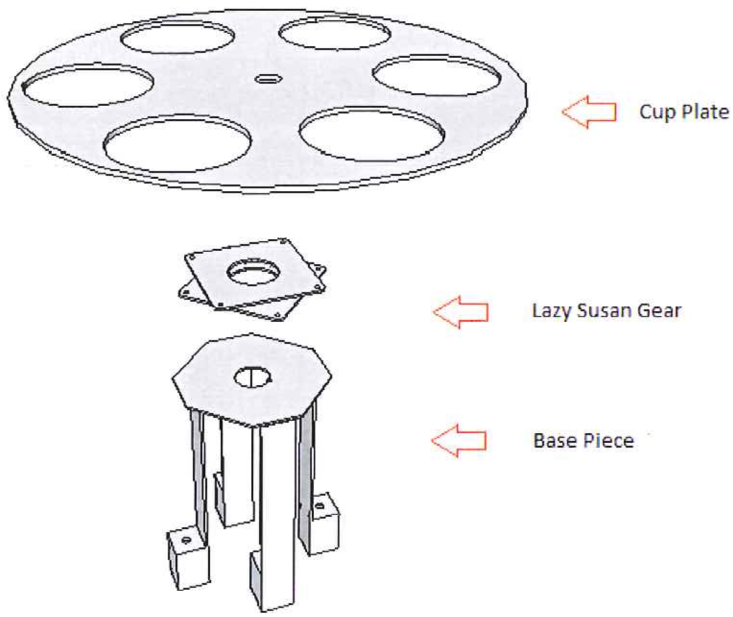


Figure 4: Turntable Subsystem Parts

The Turntable subsystem must solve the problem of transferring cups from the cup drop point, through the pouring system, and finally to the vendor and the customer. In order to satisfy the final design goals, the Turntable must:

- Transport cups to appropriate positions.
- Facilitate an average serving time of less than 1 minute per beer.
- Be able to handle a quantity of cups that is typical of an order at a festival.
- Not spill the beer during operation.
- Firmly hold the cups in place throughout operation for the pouring system.
- Allow the vendor easy access to cups while serving.
- Not increase the size of the design beyond the width of a table top.

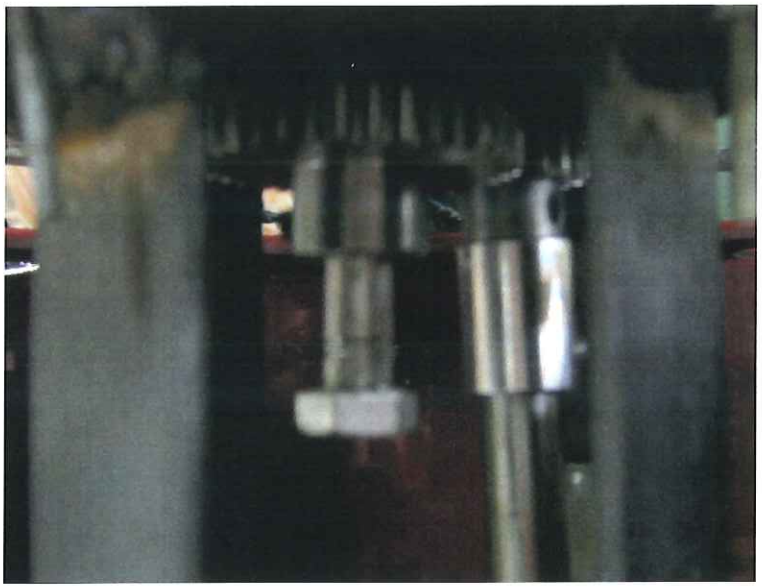


Figure 5: Turntable DC Motor Gearing

Our design was selected and refined with these criteria in mind. The final iteration of the turntable was selected primarily for its compact and robust design. This design used a thin metal disk which rotates about its center to transport the cups. The disk is attached to the Lazy-Susan in order to achieve rotational motion. After the cup is dispensed it sits in one of the circular holes of the thin circular disk, which are spaced evenly, sixty degrees apart. These cup-holes are placed the same distance from the center of rotation of the metal disk. A maximum number of six cups can be dispensed for one order until the cup-plate needs to be emptied. This quantity was selected as the maximum number of cups per order since this is well above the expected order size (1-3 beers) and fits into the space restriction without slowing down the average serving time. The Turntable design is compact and confines the movement of the cups to a circular path. This circular path takes less table space than a solution that would transport the cups in a linear fashion. One benefit of holding the cups suspended in the turntable is that this eliminates tipping once the cups have been dropped. Sloshing is the only factor that affects the maximum speed at which the turntable can safely rotate. This limit can be discovered by adjusting the voltage sent to the DC motor (See Figure 16) used to drive the cup-plate from underneath.

Not enough

The Bill of Materials lists items used for the turntable design can be seen in Appendix B.

2.3. Pouring System

The purpose of the Pouring System is to efficiently transfer beer from the keg to the cup. After the cup is dropped and positioned below the Pouring Tower, the solenoid valve then opens and fills the cup. Operational goals of the Pouring System consist of:

- Transport beer in fluid lines without leaks.
- Easily attachable to kegs with CO_2 systems.
- Angled in such a way to minimize excess foam during pour.
- Releasing liquid when a cup is underneath the spout and at no other time.

The mechanical aspects of the Pouring System design include the fluid line, solenoid valve and pouring tower. These parts are assembled in Figure 4, Figure 6, and Figure 7.

Figure 6: Solenoid Valves and Internal Fluid Lines

Figure 7: Pouring Tower

Figure 8: Tube Plate

To supply the device with beer, a user plugs a beer line into an input fluid nipple. It is expected that the line be pressurized within the standard range of twenty to twenty-five psi. These nipples connect the keg lines to another set of fluid lines that are located inside the acrylic box. These nipples, seen in Figure 8, allow for easy connection during setup. Inside the acrylic box, fluid lines are attached to the inside end of the nipples and connect to the solenoid valves. At the output of the solenoid valves, another set of fluid lines run up the Pouring Tower and come out of the tower at an angle to pour the beer down the side of the cup. This can be seen in Figure 6. The purpose of angling the tubes is to minimize any undesired excess foam. In order to transport beer through fluid lines without leaks all connector pieces are held down with metal clamps to ensure fluid lines stay intact. Testing of the Pouring System with the microcontroller is done to determine an optimal time the solenoid valve should remain open. Implementing this optimization should prevent spills.

For reconstruction and price purposes refer to the Bill of Material found in Appendix B-1. Note that this system is capable of connecting to three separate taps. The system consists of the following items:

- 6 long bolts (for valves)
- 3 solenoid valves
- Micromatic tubing (.25 inch inside diameter)
- 3 double nipple connectors (held in place by 3 rubber grommets)
- 6 valve nipple connectors
- 9 circular metal clamps
- Metal stand (Figure 9)
- 11 inches of 2" diameter PVC
- 2 – 90 degree bends of 2" diameter PVC
- 1 – PVC cap 2"
- 1 bolt and nut (for the PVC to metal stand)
- 2 bolts and nuts for the metal stand
- 1 Tube Plate(Figure 8)

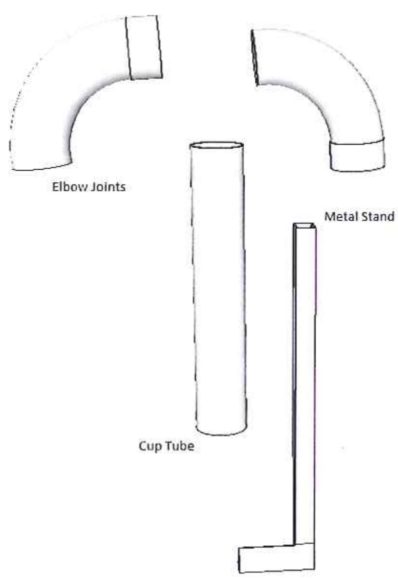


Figure 9: Pouring Tower Assembly

These items' connections are shown in Figure 6, Figure 7 and Figure 10. The height of the valves should be close to the height of the Tube Plate which holds the input fluid lines. If these lines are not kept level the tubing will pinch or excess tubing will be necessary to prevent the impedance of flow. All electrical aspects of the Pouring System are covered in the next section about the Electronics of the design.

Figure 10: Pouring System Connections

2.4. Electronics

The electronics compose the "nervous system" of the design. They are responsible for environmental data detection and control of the automated process of pouring beer. They consist of two major components the Microcontroller and the Power Circuits. The Microcontroller handles the operation software while the Power Circuits makes sure the correct voltages are delivered to corresponding subsystems.

2.4.1. Microcontroller

A PIC32mx microcontroller is used to provide controlling logic for the device. The microcontroller accepts input from the operator and from peripheral devices that the design consists of, and based on them determines the course of action. The PIC32mx microcontroller has a sufficient amount of ports in the interest of accomplishing the design goals. Among these are the analog, digital and power-width-modulated input-output ports. The microcontroller also has an input-capture, output-compare and change-notice mode for some digital ports that are necessary for the design. This microcontroller was chosen for its diversity and speed, which allows for future upgrades without sacrificing some of its functionality. Such upgrades could be, but are not limited to, an LCD display, wired and wireless network, a file system and sound support. Since several units of this design can be used during festivals and events, the possibility of networking and some other upgrades was accounted for during hardware selection. Another reason for choosing this microcontroller is that it comes with a development environment that has a full C language support and set of macro functions that make programming less tedious.

The microcontroller requires an expansion board that will map all of its ports. The PIC32mx I/O Expansion Board is used to fan out the microcontroller ports in an accessible and convenient manner. The Expansion board also supports 9 to 15 DC voltage input in order to power up the microcontroller when the USB power is not present. It also has hardware that allows miniSD storage card access as well as RJ45 LAN wire socket. Devices such as a 4x4 matrix keypad and an optical encoder, found on the Turntable motor, do not require extra hardware and therefore are connected to the microcontroller directly. On the other hand, the cup detector, solenoid valves and motors used in the design need extra circuitry in order for the microcontroller to be able to control them. The microcontroller regulates these electronics through a power regulator circuit that was specially built for this project (See Figure 11).

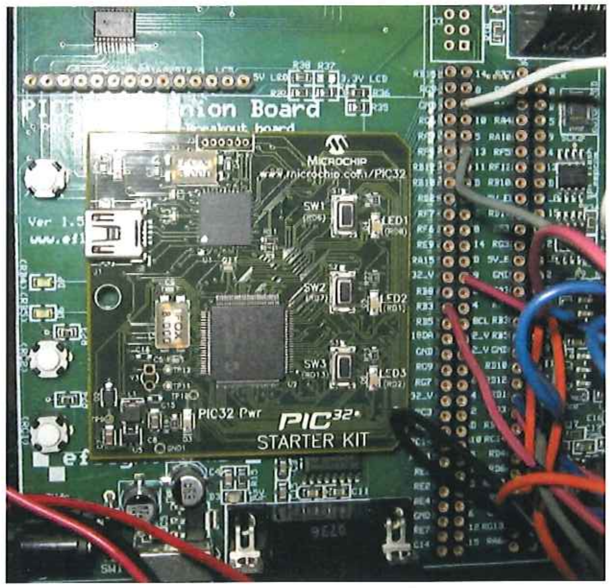


Figure 11: PIC32mx Microcontroller and Expansion Board

2.4.2. Power Circuits

A custom made power amplification circuit provided the energy necessary for some of the electronic devices. For example, the solenoid valves are controlled using the circuit shown in Figure 12. Microcontroller ports RD0, RD1 and RD2 are each connected to similar circuit shown below, which is itself connected to solenoid valves 1 to 3 respectively. The purpose of this circuit is to open and close the solenoid valve.

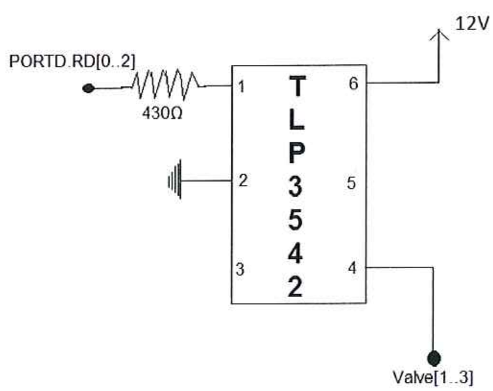


Figure 12: Solenoid Power Circuit

Five solid state relays (model #: TLP3542, see Appendix F) are used in the project (See. A relay is an electronics device that acts like an electronic switch. According to the manufacturer's specs, the solid state relay requires a voltage of 1.33VDC and 3 to 30mA of current in order to pass through up to 60 volts and 2.5amps. Equation X is used to calculate the resistor value needed to limit the supply voltage to the chip to 1.33V and a current of 5mA if 3.33V is used for controlling.

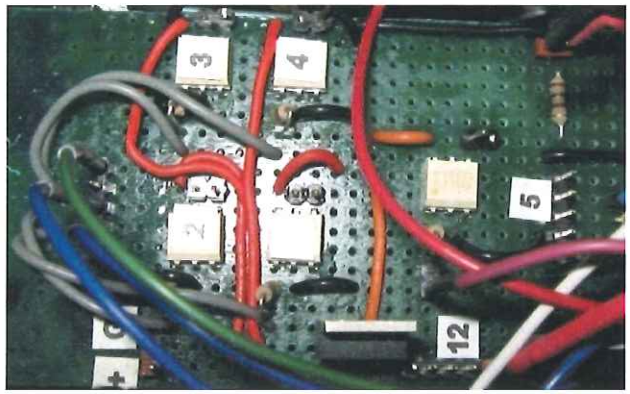


Figure 13: Solid State Relay Circuits

According to Equation 1, a 400Ω resistor will give the correct control voltage and current for this particular relay. However, this is a theoretical value. The actual resistor used was a 430 ohm resistor because of its availability. It was also selected because it met the required voltage and current parameters. The same circuit shown in Figure 12 is used in order to control the Turntable motor and Cup Dispenser motors. The Cup Dispenser and Turntable subsystems use identical power circuits to control the motors. Schematics corresponding to these configurations are available in Appendix F.

$$R_1 = \frac{(3.33 - 1.33)V}{5mA} = 400\Omega$$

Equation 1: Resistor Value in Solid State Relay Power Circuit

2.4.3. Keypad

A 4x4 matrix keypad is used in order to allow the vendor to control the device. The keypad can be seen in Figure 14. The keypad has a 4x4 matrix of wires underneath the buttons. When a buttons is pressed a connection is made in this matrix. The row pins are connected to output ports RB0, RB1, RB2 and RB3, while column pins are connected to the input ports RB4, RB5, RB13 and RB9 respectively. By applying a voltage to the output ports and measuring the voltages of the four input ports the microcontroller can determine what key is pressed and for how long. The keypad allows the vendor to enter, execute or cancel an order. The vendor can also calibrate the device pouring times using the keypad.

Figure 14: 4x4 Matrix Keypad

All software that controls the electronics is written in MPLAB IDE v8.40, compiled and burned to the microcontroller. The software is written using the C language. Appendix D has the complete code that was compiled and used for the final design. The software accepts user input from the keypad and acts accordingly. It has two modes or 'menus': Main and Configuration. In the Main Menu, a user inputs the order. The keys 1 through 3 correspond to the three valves which control fluid flow through the lines. This allows the user to select the type of beer they want by preparing a specific beer line for pouring. The number of beers the user wants can be input after a type of beer is selected. The 4 key corresponds to 1 beer and the 9 key corresponds to 6 beers. For example; pressing key "1" and then pressing key "5" will enter 2 beers for Valve 1. The user can continue entering orders until it reaches the maximum capacity of six cups. The software will automatically check and will not allow the user to enter more than six cups of beer per order. Once an order is entered, vendor can press the A key to confirm and execute the order or the B key to cancel the order. Any wrong combinations of keys will be ignored by the microcontroller. Once an order is accepted, the microcontroller will search for an empty spot in the Turntable and dispense a cup there. Afterwards, it will rotate the cup under pouring unit and pour beer into it using the pre-configured time.

The vendor can also press the "D" key to enter Configuration Menu and "C" to return to the Main Menu. In the Configuration Menu, the vendor can press keys "*", "0" or "#" to pour beer from Valves one, two or three. These keys allow the user to change and reset the timer used on each beer line effectively controlling the amount of beer poured per cup on that line. Once the key is released, the microcontroller will stop pouring and the time the key was depressed will be saved in memory for that valve.

3. Performance Testing

The proof of concept experiments addressed the machine's performance with respect to the design constraints. These experiments demonstrated the solution to the design problem of each individual subsystem and the system as a whole. The full system test was performed in order to assess the device's feasibility as a solution to the design problem.

3.1. Cup Dispenser Proof of Concept

In order for the cup dispenser to succeed in its task, it must store cups for use and dispense a single cup at a time into a cup holder. The experimental setup consisted of three main components: 1) a tall metal cylinder used to hold a stack of cups, 2) a stationary vertical steel support which holds the metal cylinder in place and 3) an aluminum disk with 6 identical holes each approximately three inches in diameter which will receive dropped cups. These components can be seen in Figure 15.

Figure 15: Cup Dispenser Equipment Setup

The metal cylinder is adjustable in diameter and is designed to hold cups stacked one inside the other. At the bottom edge of the cylinder, two vertical threaded "screws", attached to DC motors, are fastened on opposite sides of the cylinder (See Figure 2). The distance between the inside diameters of the screws is equal to the width of a cup. This is so the fall of the cup is controlled by the turning of the screws. The DC motors were controlled manually for this experiment because the microcontroller was not programmed at that time. The experiment was designed to test if the complete setup could drop a single cup in a hole on the turntable using manually controlled voltages for the screws. For the cup dispenser apparatus to be a feasible solution, a single cup must fall when the DC motors are provided with a voltage.

3.2. Turntable Proof of Concept

The third system, the Turntable subsystem, consists of a cup plate, a disc of aluminum with holes punched in it to act as cup holders, rotating freely on a lazy Susan gear, supported by four steel legs attached to a ridged surface beneath the apparatus. The whole setup can be seen in Figure 16.

Figure 16: Turntable Experimental Setup

A DC motor is held to a gear attached to the turntable from underneath. The motor has an optical encoder that outputs a digital high signal five hundred times per revolution. These pulses can be used as position and velocity control by a microcontroller. Two simple tests were performed: one to prove the DC motor's optical encoder is functioning properly, and one to prove that a DC motor can rotate the table while in operation. The sloshing factor was deemed not an issue because the twelve volts supplied by the microcontroller to the Turntable DC motor did not create enough angular acceleration to upset any liquid from the cups. For the first test, power needs to be supplied to both the DC motor and its optical encoder. An oscilloscope was used to monitor the data lines coming out of the optical encoder. The pulse signal that would allow for angular position and velocity control was found on the green wire, thus proving the signal is usable. For the second test, the DC motor was fastened to one of the legs underneath the turntable, so that the gears of the motor and Turntable are engaged with each other. The experiment's goal is to determine if the motor provides enough torque to rotate the cup plate on the Lazy Susan gear.

3.3. Pouring System Proof of Concept

Control of the volume of fluid poured to within ±0.5 fluid ounces, is the goal for the pouring system's proof of concept experiment. The set up for the pouring system consists of a tank, a solenoid valve that is controlled by a microcontroller, and a graduated cylinder. Water was used as the working fluid for this experiment. The tank was pressurized with a hand pump to 22 psi because this was within the industry standard operating range. The high pressure in the tank provided a force on the liquid in the tank. This forces the fluid up through the tube exiting the tank at some rate. The tube runs out of the tank to a solenoid valve that controls the fluid flow. This valve is connected to a microcontroller, which opens and closes the valve. Once 12 volts are applied, the liquid flows through the valve, up and out the end of the tubing into a graduated cylinder for measurement. The experiment is meant to prove it is possible to predict and pour a set amount of liquid using only a timer. The microcontroller is programmed to supply a current to the valve for a set time period. The microcontroller has a built in clock which functions as a stop watch counting in 25 nanosecond intervals. The theory behind the setup is simple. The system pours a volume of fluid during the time that the solenoid is energized. The longer the pour time, the more fluid is dispensed. It was thought that using an accurate timer to control the length of a pour would produce the desired result. The purpose of this proof of concept test is to verify this hypothesis. It is important to note the pressure for the initial pouring test was not constant, thus linearity between time and volume cannot be assumed. Another experiment, with the same setup was used to obtain more data in order to prove linearity between volume and time for a non-constant pressure system. If the data stays relatively linear, with respect to volume poured and time, linearity may be assumed for further testing.

3.4. Pouring System Accuracy Testing

This experiment was setup and tested in order to confirm the system is accurate and that foaming is not be an issue. The only functional difference between this experiment and the proof of concept is the addition of a pressure controlled chamber. A constant pressure would allow for the assumption of a linear relationship between volume poured and time, in turn allowing us to take data to calibrate the timing with respect to volume poured. Twelve runs total, two runs for each time interval, are conducted in the experiment. With this information, it is possible to understand the relationship between the volume of liquid poured and time. This information can be analyzed to predict the pour times for different volumes of liquid.

3.5. Full System Testing

In order to confirm that the design satisfies the criteria set out at the beginning of the year, a whole system test was performed. All three of the subsystems were hooked up to a microcontroller. Once programmed the microcontroller would act as the brains of the machine's operations. The cup dispenser is positioned over a single hole on the turntable. The pouring system was positioned over another hole of the turntable. There are three tubes that can dispense beer. All three lines are positioned over a single cup hole so that all three pour into the same cup. Three solenoid valves control the flow of beer through these fluid lines, which means the user can choose between three different types of beer. The cup dispensing and pouring subsystems are attached to towers offset sixty degrees clockwise respectively around the edge of the turntable. The two towers are protruding from an encasement of acrylic that houses the solenoid valves, electronics and tubing. This case protects the delicate equipment inside from spillage. The full system test will use the completely assembled automated machine.

The system will drop a cup into a turntable and rotate the cup under the pouring system, where the cup will be filled with beer. When the cup is full it will rotate around for delivery to the customer. Goals of this test are: that this system is capable of delivering a beer within a minute of the order being entered, the system can accurately pour a beer, drop a cup and maintain correct position with the turntable, and that the quality of the beer is maintained from the keg. Good quality beer means it is lacking in any unwanted particulates is not flat and contains a healthy amount of foam (approximately 10% of the total volume). If the design can meet this constraint then it is a feasible solution to the design problem.

New

Unfortunately, prior to the full system test, a live wire fell onto the microcontroller and burnt the chip. Due to these events, the full system test was broken into two parts. These were separate tests for the system software and hardware to verify the functionality of each. The two experiments assume the full system will function when the hardware and software are integrated. The final fully integrated system experiment was not performed because a new microcontroller could not be acquired in time.

3.5.1. Hardware Testing

Having tested Cup Dispenser, Turntable and Pouring subsystems individually, the final hardware experiment consisted of all three operated in the proper order to serve a cup of beer. The hardware experiment was a completely hand controlled process. Since the microcontroller was burnt, a person manually ran the motors. The order for the experiment is as follows: Cup motors are initiated first in order to drop a cup, followed by the Turntable motor to change the cups position, then the solenoid valve to fill the cup and finally the Turntable motor rotates again to bring the cup around for delivery. For this system test a CO2 tank pressurized the fluid lines. The fluid lines were pressurized at twenty-two psi by a CO2 tank. Some foaming was observed in the beer lines not present in the previous pouring system experiments. The tubing used in this experimental apparatus had an inner diameter significantly less than a quarter inch. All previous Pouring experiments used a quarter inch inner diameter tube as that is the industry standard for large events where beer is sold. As the inner tube diameter decreases, while maintaining constant pressure, the Reynolds's number for pipe flow increases, which can produce turbulent flow and decrease the quality of the beer. This causes excessive foam, so the pressure for the experiment was reduced to 5 psi. An increased pour time is expected because of this low pressure in the beer lines.

3.5.2. Software Testing

The software experiment was implemented in several parts. Each part of the software that controls a particular subsystem is regarded as a module and is tested separately prior to integrating it into the final software. These components include the code for the keypad and the three mechanical subsystems: Cup Drop, Turntable, and Pouring System. The keypad module was tested by hooking all of its pins into a microcontroller which could detect what buttons were pressed and for how long. A small code was added to the module that would print out the pressed key information detected by microcontroller to the debugging screen. This information includes the binary code passed by the keypad driver, as well as key interpretation by microcontroller and the time the key was depressed. Once it is verified that the microcontroller can detect key presses correctly, the timing ability of the module is tested. A key was pressed four different times, varying from 10 second up to 3 minutes. The length of time a key was pressed for was timed by digital stopwatch and compared to the value printed out by the microcontroller. Since the stopwatch was operated by a human, the timing precision for the test was set to be 1 second in order to account for human error.

In order to test the Cup Dispenser module, the microcontroller was connected to the Cup dispenser motors. Some code was added to the module so that the Cup Dispenser module was activated using one of the keys on the keypad. Cup presence was detected using Infrared Detector. When the cup successfully drops, it reflects the beam of the IR detector and the microcontroller registers this as a digital high on the RG0 input port. Once this signal is received the microcontroller knows to stop the Cup Dispenser motors, completing the action of dropping a cup. The functionality of the IR circuit was tested before experimentation using a voltmeter and a NIOSA cup: this was done to avoid possible error during testing.

The Turntable module was tested using a signal generator, one of the microcontroller's onboard LEDs and the information output to a debug screen. A square wave of different frequencies was generated using the signal generator and applied to RG8 and RG9 ports. RG8 is used to track the position of the Turntable while RG9 is used to detect a full revolution of the turntable. Signals for both ports are provided by an optical encoder in the final design.

In the module test for simulating the Pouring system, the microcontroller was configured to open the solenoid valve for a specific amount of time. During the initial testing, an LED was used instead of valves. The pour time was varied between 4 and 60 seconds. The timing accuracy was verified using a stopwatch and again 1 second precision was used for accuracy.

4. Results

A detailed analysis of the results for each component and the final design testing will determine if the design is validated. The results are validated by showing that this design will increase a vendor's serving speed and efficiency at events and festivals. It will also explain how these results demonstrate this validation.

4.1. Cup Dispenser Proof of Concept Results

From the results of the experiments and the data gathered, it is possible to evaluate the capabilities of the prototype and design in general. For each system, data was taken in order to assess the design's capabilities. The cup drop experiment simply tested the subsystem's ability to drop a cup into a hole repeatedly. With this respect, the design performed well. The apparatus performed exactly as predicted. When the DC motors were supplied with a voltage, they rotated, turning the two cup screws on either side of the metal tube they are mounted on, and a cup fell into a hole in the cup plate. The test was repeated multiple times, all ending in a successful drop. This proves the cup dispensers can drop a single cup repeatedly into a hole. For the whole system to be automated, the Pouring System and Turntable experiments should show similar results of reliability. The speed of the DC motors is controlled by varying applied voltage between five and twelve volts. A significant increase in motor speed was noticed which led to faster drop times. At higher voltages, the motor emits a small amount of audible noise. Extensive testing was not performed because during the initial tests it was determined that five volts would be sufficient for the purpose of dropping a cup quickly. The higher applied voltages were avoided because they produced too much noise.

4.2. Turntable Proof of Concept Results

The experiment to test the cup transportation subsystem consisted of controlling the turntable via microcontroller. After programming the microcontroller to utilize the pulse signal coming from the motor's optical encoder, it was possible to measure the angular position and speed of the turntable. Given this basic information, the microcontroller calculated the voltages necessary to drive the motor. There was some concern the voltage provided to the motor would provide a large angular acceleration, causing liquid to slosh out of a cup and fall onto the motor. In order to counteract this, a control scheme for the motor should not allow it to accelerate to a point where spilling occurs. The microcontroller has a very simple control scheme for the motor; it was set up to supply either twelve or zero volts to the motor. These twelve volts did not provide enough power to upset any liquid, because of the motor's gear ratio. This proves the Turntable's ability to transport cups through the process of pouring a beer without spills.

4.2. Pouring System Proof of Concept Results

The pouring system was tested twice. The first experimental setup used a pressure chamber that provided a non-constant pressure curve with respect to time while pouring. The results of the experiment can be seen in Figure 18.

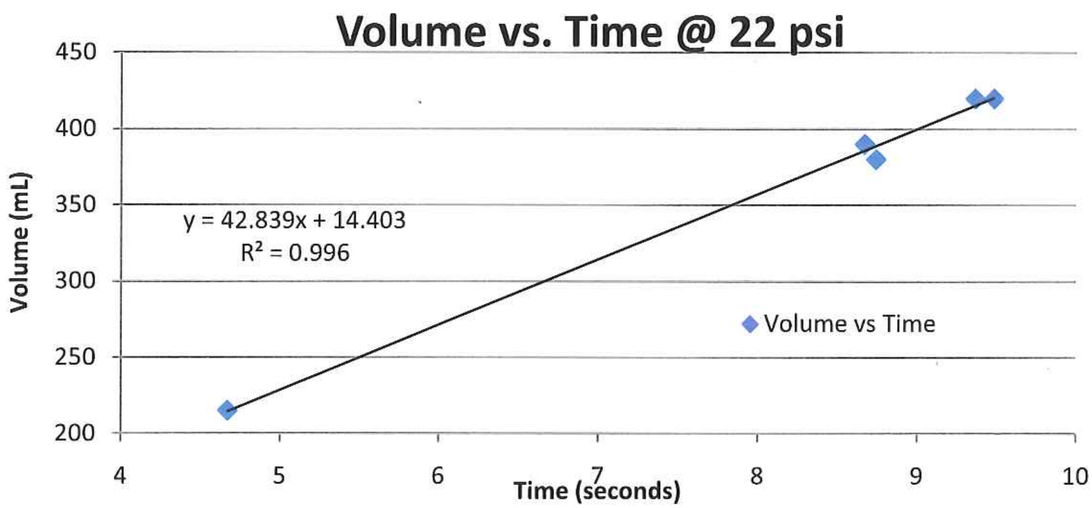


Figure 17: Volume Poured Versus Time of Pour Experiment One

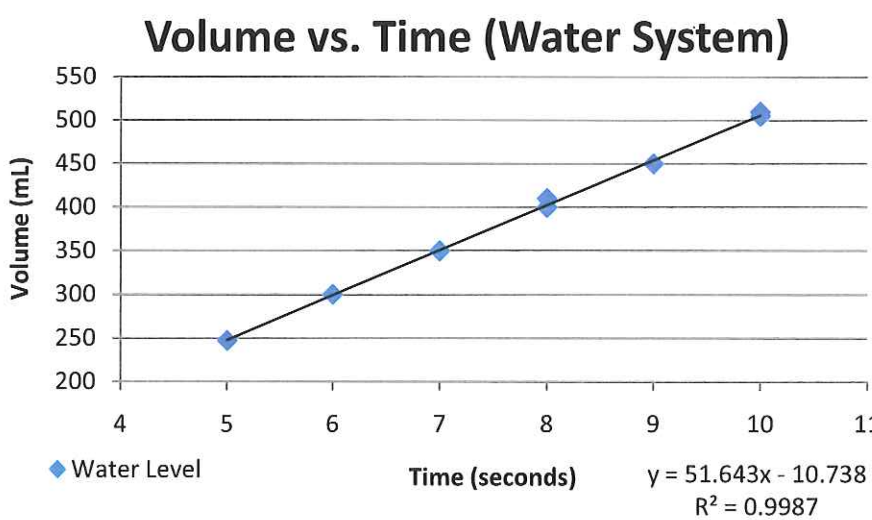


Figure 18: Volume Poured Versus Time of Pour Experiment Two

In this experiment, the apparatus performed remarkably well. All of the runs taken fell within 1mL of the predicted volume. The experiment proved a linear relationship between volume poured and time of pour. Based on this, it was possible to pour a volume of fluid accurately.

It is important to note that because the pressure chamber used in the first experiment did not provide a constant pressure it is necessary to perform a second test in order to confirm the results of the first test. It was found that the system was capable of producing consistently accurate volumes based on an interval of time using a non-constant pressure chamber. The results of the second test are in

4.4. Accuracy Test Results

The purpose of the second experiment for the pouring system was the same as the first: to control a poured volume of fluid using only a timer. Beer was used as the working fluid for this experiment, because of this foaming was monitored. A head of ten percent of total volume is considered a healthy amount, so this was the standard used to judge the individual pours. The second experiment used a pressurized chamber that provided a linear pressure vs. time relationship. Results from this test showed linearity within the range of 20 and 25 psi. As the system stretched far outside these boundaries, the volume versus time became far less linear and tended to look exponential. It was confirmed that the volume of beer did change linearly with respect to time in the constant pressure setup. Considerably less foam was produced in all of the test runs of this second experiment, as compared to the first. This is a result of the system pouring into a NIOSA cup not a graduated cylinder. A constant pressure provides laminar flow as opposed to turbulent flow seen in the non-constant pressure test. The foam in the previous experiment was a result of the length of the graduated cylinder and its imperfections. The accuracy tests show the design can use only a timer to pour a specified volume of good quality beer. This proves that the system will distribute a desirable beer that is not lacking in carbonation. However it is important to maintain laminar flow in order to maintain the quality of the beer.

4.5. Final System Testing Results

4.5.1. Hardware

The hardware testing consisted of the testing of the three major subsystems in the proper order. Upon applying voltage to the DC motors attached to the cup dispenser a cup was successfully dropped into the Turntable. This was done with five and twelve volts, their approximate times were five seconds and one second respectively. The Turntable motor was fed power and it successfully rotated the cup plate sixty degrees, lining up the cup with the pouring tower nozzle. The solenoid valve on the active beer line was then fed twelve volts and the liquid was allowed to pass. The pour lasted for approximately thirty-three seconds. The turntable was fed power again and the cup plate rotated, moving the cup out to be picked up. The total process took forty-nine seconds. This proves that the design can successfully dispense a cup of beer in less than one minute.

4.5.2. Software

During Keypad module test different keys were pressed in different sequences and the microcontroller detected all key presses correctly and printed the corresponding key information on the debugging screen. It was found that the microcontroller timed key presses successfully for both extremes (10 seconds to 3 minutes) within 1 second of precision. This test verified that microcontroller can successfully detect keys pressed on keypad as well as time how long it was pressed.

During Cup Dispenser module test it was found that the microcontroller stopped the motors every time a digital high was applied to port RG0 from the IR detector. This proves that Cup Dispensing module can successfully detect cup drop and stop the motors.

During the Turntable software module testing, the microcontroller successfully counted digital highs generated by the signal generator for different square wave frequencies and applied to table position tracking port RG9. This proves it is possible to control the motor using only the output signal from the motor's optical encoder. The corresponding count was outputted to a debugging screen so that the devices operation could be verified visually. Also every time digital high was applied to port RG9, the count was set to 0. This verifies that microcontroller can detect full revolution of the turntable and set the position count back to 0.

During the Pouring module test, it was found that the microcontroller was able to keep the LED On for given amount of time within a 1 second precision. This module was later used in the Pouring Proof of Concept test with actual valves and liquid. The module performed perfectly during those test and was able to control power to the valves correctly. Once again, this proves that the module can control valves and keep them open for set amount of time.

5. Conclusions and Recommendations

The design performed very well with regards to the design goals. It is able to serve a beer in less than one minute with no excessive foam, and was able to pour an accurate amount of beer. The final product cost less than one thousand dollars which is an economical solution for the target audience: vendors at public events and festivals. The machine delivers cold refreshing beer, without particulates. The design conforms to all Texas State laws, and is easily movable. The keypad provides a user friendly control scheme that allows for user control of serving size for each cup.

There are some aspects of the design that can be improved by a future design group that works on the Automated Beverage Dispenser. The microcontroller can only perform tasks sequentially. This means it can only perform one task at a time. As a result the design can fill a cup, but cannot dispense another cup until it is finished pouring. This sequential process lengthens the time required for the machine to finish pouring an order. A future group could program the microcontroller to carry out multiple processes in parallel, in order to reduce the pour time. It is recommended that any future groups that work with a microcontroller on this type of project take the time to investigate this alternative. Another consideration for future work is the motor attachment. The DC motor that rotates the turntable is fastened onto a leg of the table itself. It is oriented at a ninety degree angle to the horizontal. As a result, it is difficult to reach the motor in order to make alterations to its orientation. A horizontal orientation was not used because it required the turntable gear to have a longer gear shaft. A component of these proportions could not be found because of the specific nature of the problem. A custom piece was too expensive a solution, and the vertical orientation solution was stable enough for the purposes of the design. A permanent stand or holder for a motor attached to a ridged surface would prove to be an easier solution, at the cost of expense. At the beginning of the fall semester of this project, one of the proposed solutions for a user interface was a small color LCD display. However, the screen was very difficult to program and was not necessary for the completion of the design goals.

Overall, the final product satisfied all of the design goals except repeatability. This experiment could not be performed due to an accident during setup in which the microcontroller being used was burnt. If a replacement microcontroller could be obtained, a test of the final product's repeatability is all that is left to do.

Appendix A

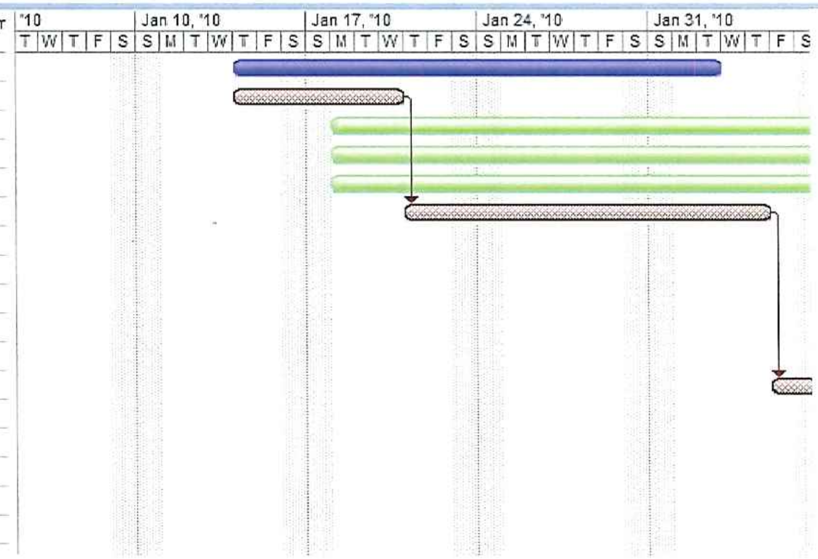
WBS

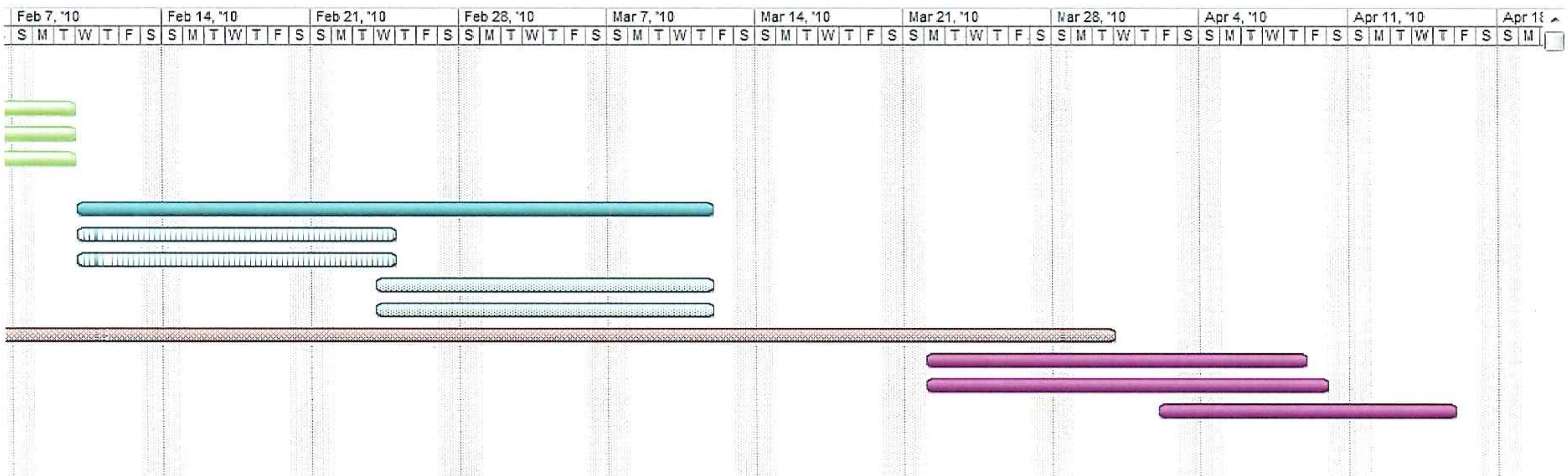
1. Project Work
    1.1. Initiating the Project
        1.1.1. Problem Statement
            1.1.1.1. Library Research
            1.1.1.2. Brainstorming
            1.1.1.3. Writing/Editing
        1.1.2. Charter
            1.1.2.1. Review Doc Spec
            1.1.2.2. Define Scope
            1.1.2.3. Define Requirements
            1.1.2.4. Writing/Editing of Charter
            1.1.2.5. Meeting with Sponsor
        1.1.3 Problem Description Presentation
    1.2. Initial Design
        1.2.1. Literature Review / Investigation
        1.2.2. Brainstorming Approaches
        1.2.3. Analyzing/Testing Potential Approaches
        1.2.4. Documentation
            1.2.4.1. Design Matrix
            1.2.4.2. Design Review Presentation
            1.2.4.3. Design Report
    1.3. Prototype/Proof of Concept
        1.3.1. Specifying Functionality
            1.3.1.1. Cup Dispenser
            1.3.1.2. Pouring System
            1.3.1.3. Turn Table
            1.3.1.4. Microcontroller
        1.3.2. Ordering Parts
        1.3.3. Construction
            1.3.3.1. Cup Dispenser
            1.3.3.2. Pouring System
            1.3.3.3. Turn Table
            1.3.3.4. Microcontroller
        1.3.4. Testing/Evaluation
            1.3.4.1. Cup Dispenser
            1.3.4.2. Pouring System
            1.3.4.3. Turn Table
            1.3.4.4. Microcontroller/Debugging
        1.3.5. Documentation
            1.3.5.1. PPOC Demonstration
            1.3.5.2. PPOC Presentation
        1.3.6. Microcontroller Reading
    1.4. Final Design
        1.4.1. Specifying Functionality
        1.4.2. Ordering Parts
        1.4.3. Construction

            - 1.4.3.1. Cup Dispenser
            - 1.4.3.2. Pouring System
            - 1.4.3.3. Turn Table
            - 1.4.3.4. Microcontroller
        - 1.4.4. Testing/Evaluation
            - 1.4.4.1. Efficiency
            - 1.4.4.2. Serving Time
            - 1.4.4.3. Reload Time
            - 1.4.4.4. Accuracy
            - 1.4.4.5. Quality
            - 1.4.4.6. Final (Full System)
        - 1.4.5. Documentation
            - 1.4.5.1. Bill of Materials
            - 1.4.5.2. CAD Drawings / Assemblies
            - 1.4.5.3. Final Report
            - 1.4.5.4. Final Presentation
    - 1.5. Closeout
        - 1.5.1. Cleanup of area/project
        - 1.5.2. Clearance Form
- 2. Administrative
    - 2.1. Planning
        - 2.1.1. Work Breakdown Structure
        - 2.1.2. Schedule
        - 2.1.3. Budget
        - 2.1.4. Project Plan Writing/Editing
    - 2.2. Project Management
        - 2.2.1. Monthly Management Reviews
        - 2.2.2. One-on-Ones w/Dr Nickels
    - 2.3. Self-Peer Evaluations
    - 2.4. Group Meetings
    - 2.5. Executive Summary
- 3. Course Content (Non-Project)
    - 3.1. Reading
    - 3.2. Studying
    - 3.3. Homework/Quizzes
    - 3.4. In-Class time

Appendix A: Gantt Chart

	Task Name	Duration	Start	Finish	Predecessors	Resource Nam
1	Revise Project Plan	14 days?	Thu 1/14/10	Tue 2/2/10		
2	Programming: High Level Design	5 days?	Thu 1/14/10	Wed 1/20/10		
3	Proof of Concept: Cup Dispenser	17 days?	Mon 1/18/10	Tue 2/9/10		
4	Proof of Concept: Pouring System	17 days?	Mon 1/18/10	Tue 2/9/10		
5	Proof of Concept: Turn Table	17 days?	Mon 1/18/10	Tue 2/9/10		
6	Programming: Coding (Proof of Concept)	11 days?	Thu 1/21/10	Thu 2/4/10	2	
7	Testing: Efficiency	22 days?	Wed 2/10/10	Thu 3/11/10		
8	Testing: Serving Time	11 days?	Wed 2/10/10	Wed 2/24/10		
9	Testing: Reload Time	11 days?	Wed 2/10/10	Wed 2/24/10		
10	Testing: Accuracy	12 days?	Wed 2/24/10	Thu 3/11/10		
11	Testing: Quality	12 days?	Wed 2/24/10	Thu 3/11/10		
12	Programming: Testing & Debugging (Revis	38 days?	Fri 2/5/10	Tue 3/30/10	6	
13	Final Assembly	14 days?	Mon 3/22/10	Thu 4/8/10		
14	Testing: Final	15 days?	Mon 3/22/10	Fri 4/9/10		
15	Final Report Draft	10 days?	Fri 4/2/10	Thu 4/15/10		





Appendix B: List of Vendors and Bill of Materials

Turntable		
Part (PN) (Company)	**Metal**	**Price**
Gear Shaft	Steel	$1
Pulleys (6z53m084sf0910 & 6z53m032df0906) (SDP/SI)	NA	$20
Lazy Susan bearing (#02z21) (Woodcraft)	NA	$3
Cup plate	Aluminum	$1.40
Legs	Steel	$4
Base Plate	Steel	$2
DC Motor (ROB-09238) (Sparkfun)	NA	$15
Miscellaneous	NA	$5
DC Motor Driver (ROB-09402) (Sparkfun)	NA	$15
Total		**$66.40**

Microcontroller/Electronics	
Part [PN] (Company)	**Price**
Power Supply [Altech #PS-S6012] (Power Supply Dirtect)	$57
Solenoid Valves [72R9DGV-12VDC] (PeterPaul)	$80x3=$240
Servos [900-00005-ND] (DigiKey)	$13x2=$26
PIC32mx [DM320001] (Microchip)	$50
PIC32 I/O Expansion Board [DM320002] (Microchip)	$72
Graphics PICtail Plus Daughter Board [AC164127] (Microchip)	$135
Voltage Regulator [LM340T-5.0-ND] (DigiKey)	$1.74x4=$6.96
Total	**$587+shipping**

Cup Dispenser	
Part (PN) (Company)	**Price**
Cup Screw	$5
Cup Holder (Suite Supply)	$15.26

DC Motor & Driver (Pololu Electronics)	$15
Total	**$35.26**
Pouring System	
Part (PN) (Company)	**Price**
Valves	Price listed in Electronics section
Tubing (548C)	$0.90/ft * (~10ft)
Insulation (sku#420504)(Home Depot)	$5.77 (6 ft)
Total	**$25**

Appendix C: Budget

Date Submitted 4/14/2010
Submitted By: Iuri G.
Group Name: Moody Loody
Advisor Name: Dr. Kelly-Zion

Income

Date	Sponsor	Description		Status	Budgeted Amount	Actual Amount	Notes
9/2/2009	Engr Dept	Senior Design Project Allotment			$1,200	$1,200	

Total Income $1,200 $1,200

Expenses

Date	Vendor	Item Description	PO #	Status (Planned/ Pending/ Cleared)	Budgeted Amount	Actual Amount	Status (Check one): Internal	Dept Purchase Order	PCARD	Reimbur-sement	Notes
11/7/2009	Home Depot	Miscellaneous (Proof of Concept)		cleared	$5.25	$5.25			x		
12/10/2009		Gear Shaft		cleared	$1.00	-	x				
12/10/2009		Pulleys	6z53m084sf0910 & 6z53m032df0906	cancled	$20.00	-	x				
12/10/2009	Woodcraft	Lazy Susan bearings	02z21	Cleared	$3.00	$8.49					
12/10/2009		Cup Plate		Cleared	$1.40	-	x				
12/10/2009		Legs (turntable)		Cleared	$4.00	-	x				
12/10/2009		Base Plate		Cleared	$2.00	-	x				
12/10/2009	Sparkfun	DC Motor	ROB-09238	Cleared	$15.00	-					
12/10/2009	Home Depot	Turntable Miscellaneous (screws, etc.)		Cleared	$5.00	-					
12/10/2009	Sparkfun	DC Motor Driver	ROB-09402	Cleared	$15.00	-					
12/10/2009	Altech	Power Supply	PS-S6012	Cleared	$57.00	$67.45		x			
12/10/2009	PeterPaul	Solenoid Valves	72R9DGV-12VDC	Cleared	$260.00	$258.90		x			
12/10/2009	DigiKey	Servos + Kepad	900-00005-ND	Cleared	$39.64	$45.13		x			
12/10/2009	Microchip	PIC32mx	DM3200001	Cleared	$50.00	$47.36		x			
12/10/2009	Microchip	PIC32 I/O Companion Board + LCD		Cleared	$200.00	$197.50				x	
12/10/2009	DigiKey	5v IC Regulator	LM340-5.0-ND	Cleared	$5.22	$10.71		x			
12/10/2009		Cup Screw		cleared	$5.00	$0.00	x				
12/10/2009	Suite Supply	Cup Holder		cleared	$15.26	$0.00					Chris Kledges purchased and did not get reimbursement
12/10/2009	Pololu Electronics	DC Motor & Driver		cleared	$15.00	$0.00					Chris Kledges purchased and did not get reimbursement
12/10/2009	Micromatic	Tubing	548C	cleared	$9.00	$24.50				x	
12/10/2009	Acroname	Line detector		Cleared	$14.95	$19.90		x			
12/10/2009	Home Depot	Insulation	sku#420504	Cleared	$5.77	$0.00					
1/20/2010	Stock Drive Products	Gears		cleared	-	$51.44					
1/21/2010	Mouser Electronics	IR Detector and Emitter		Cleared	$10.11	$10.11				x	
1/28/2010	Proof of Concept: Pouring System			Cleared	$40.00	$27.87			x		
1/29/2010	Intertex	PCB mounts		Cleared	$10.81	$10.81				x	
$143.22	Acrylic casing	Acrylic pieces		Cleared	$70.00	$68.73			x		
3/25/2010	Digi-Key	5 SS Relays		cleared	$30.06	$30.06				x	
3/30/2010	Home Depot	Pouring System Miscellaneous		Cleared	$4.64	$4.49				x	
3/31/2010	Westbrook Metals	metal plate		Cleared	$70.00	$70.00			x		
4/5/2010	Plastic Center	Acrylic molding		Cleared	$50.00	$45.00			x		
4/6/2010	Home Depot	Final Design Hardware		Cleared	$50.00	$45.76				x	
4/7/2010	Microchip	PIC32mx Starter Kit	DM3200001	Cleared	$49.99	$44.27			x		
4/9/2010	Intertex	Box, 2x25pin ports, 25pin 6ft cable		cleared	$15.80	$15.80				x	
4/13/2010	Home Depot	Various Parts		cleared	$3.22	$3.22				x	
4/13/2010	Home Depot	Various Parts				$26.89					

Total Expenses $1,153 $1,140

turntable	$12
turntable legs	5
turntable top	2
towers	20
mis hardware	10
turntable motor	50

Budget Remaining

Budgeted	Actual
$46.88	$60.36

Notes:
* Always use Trinity Tax Exempt Form for purchases
* Please submit reimbursement receipts within one week of purchase

***This is to be update as purchases happen**

		percentage:
Final Project Cost =	$979.11	#VALUE!
Structural =	$286.38	#VALUE!
Electronics Hardware =	$237.43	#VALUE!
Turntable =		#VALUE!
Pouring =	$291.11	#VALUE!
Cup Drop =	$35.26	#VALUE!
		#VALUE!

Appednix D: Software Code

C:\Users\Iuri Gagnidze\Documents\BeerBot\main.c

```
/*************************************************************************************
                              BeerBot - Main Source Code
 *************************************************************************************
 * FileName:        main.c
 * Dependencies:    None
 * Developer IDE:   MPLAB IDE 8.20 or higher.
 * Compiler:        MPLAB C Compiler for PIC32 v1.04 or higher.
 * Company:         Moody-Loody.
 *
 * Copyright (c) 2010 Moody-Loody.
 * Software License Agreement
 *
 * THIS SOFTWARE IS PROVIDED IN AN "AS IS" CONDITION. NO WARRANTIES,
 * WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED
 * TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
 * PARTICULAR PURPOSE APPLY TO THIS SOFTWARE. THE COMPANY SHALL NOT,
 * IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL OR
 * CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.
 *
 *************************************************************************************
 * Description:
 *          This program controlls BeerBot using PIC32MX uController.
 *
 * Platforms:
 *          PIC32MX Starter Kit DM320001
 *          PIC32MX USB Starter Kit DM320003
 *
 * Tools:
 *          1. MPLAB IDE 8.20 or higher
 *          2. MPLAB C Compiler for PIC32 v1.04 or higher
 *          3. General Purpose Starter Kit DM320001 or USB Starter board DM320003
 *          4. USB Cable
 *
 * Starter Board Resources:
 *      -Debugger:
 *              JTAG.TMS        = PORTA.RA0
 *              JTAG.TCK        = PORTA.RA1
 *              JTAG.TDO        = PORTA.RA5
 *              JTAG.TDI        = PORTA.RA4
 *              PGC2            = PORTB.RB6
 *              PGD2            = PORTB.RB7
 *
 *      -LCD S1602DTR:
 *              RS              = PORTB.RB15
 *              R/W             = PORTD.RD5
 *              E               = PORTD.RD4
 *              DB0             = PORTE.RE0
 *              DB1             = PORTE.RE1
 *              DB2             = PORTE.RE2
```

```
 *              DB3             = PORTE.RE3
 *              DB4             = PORTE.RE4
 *              DB5             = PORTE.RE5
 *              DB6             = PORTE.RE6
 *              DB7             = PORTE.RE7
 *
 *      -Keypad:
 *              ROW_1           = PORTB.RB0  (output)
 *              ROW_2           = PORTB.RB1  (output)
 *              ROW_3           = PORTB.RB2  (output)
 *              ROW_4           = PORTB.RB3  (output)
 *              COL_1           = PORTB.RB4  (input)
 *              COL_2           = PORTB.RB5  (input)
 *              COL_3           = PORTB.RB13 (input)
 *              COL_4           = PORTB.RB9  (input)

 *      -Turntable:
 *              MOTOR_TRACKING  = PORTG.RG9  (CN)
 *              MOTOR_RESET     = PORTG.RG8  (CN)
 *              MOTOR_SIGNAL    = PORTD.RD3  (output)
 *
 *      -Pouring:
 *              VALVE_1         = PORTD.RD0  (output)
 *              VALVE_2         = PORTD.RD1  (output)
 *              VALVE_3         = PORTD.RD2  (output)
 *
 *      -Cup Dispenser:
 *              IR EMITTER      = N/A        (from PCB)
 *              IR DETECTOR     = PORTG.RG7  (CN)
 *              CUP_SIGNAL      = PORTG.RD0  (output)
 *
 * Starter Board Notes:
 *          1.  Do not disable the PIC32MX JTAG.  This will prevent the PIC32MX Starter Kit
 *              debugger(PIC18F4550) from communicating with the PIC32MX device.
 *          2.  Do not configure the SYSTEM CLOCK to operate faster than 80MHz.
*********************************************************************************
 * Change History:
 * ID           Date            Changes
 * --------     ----------      ----------------------------------------------------
 * MAIN_001     02/10/2010      Project created.
 * MAIN_002     02/18/2010      Added Pouring driver.
 * MAIN_003     03/01/2010      Pouring driver now uses interrupts.
 * MAIN_004     03/12/2010      Pouring driver functions optimized.
 * MAIN_005     03/17/2010      Kepad driver added.
 * MAIN_006     04/02/2010      TurnTable driver added.
 * MAIN_007     04/16/2010      Some debugging and minor corrections.
 * MAIN_008     04/19/2010      Ports changed for Cup Disp. and TTable.
 *
 * TABLE_001    04/01/2010      Turntable driver created.
```

```
 * TABLE_002    04/04/2010     Ports changed. Using CNx instad of ICx.
 * TABLE_003    04/09/2010     Added definitions for position and implemented counting.
 * TABLE_005    04/13/2010     Turn Table ISR now controls serving too. Damn!
 * TABLE_006    04/14/2010     A lot of bug fixes. Accounted for gear ration 1:2.
 *
 * KEY_001      03/17/2010     Keypad driver created.
 * KEY_002      03/23/2010     Added Debounce support.
 * KEY_003      03/30/2010     Added key timing support.
 * KEY_004      04/12/2010     Added valve controll for config menu
 * KEY_005      04/15/2010     Better time resolution. accounted for several ms loss
 ************************************************************************************/

/* Adds support for PIC32 Peripheral library functions and macros */
#include <plib.h>
#include <p32xxxx.h>

//Configuration Bits
#pragma config FNOSC     = PRIPLL        //Oscillator Selection
#pragma config FPLLIDIV  = DIV_2         //PLL Input Divider (PIC32 Starter Kit: use divide by 2 only)
#pragma config FPLLMUL   = MUL_20        //PLL Multiplier
#pragma config FPLLODIV  = DIV_1         //PLL Output Divider
#pragma config FPBDIV    = DIV_1         //Peripheral Clock divisor
#pragma config FWDTEN    = OFF           //Watchdog Timer
#pragma config WDTPS     = PS1           //Watchdog Timer Postscale
#pragma config FCKSM     = CSDCMD        //Clock Switching & Fail Safe Clock Monitor
#pragma config OSCIOFNC  = OFF           //CLKO Enable
#pragma config POSCMOD   = XT            //Primary Oscillator
#pragma config IESO      = OFF           //nternal/External Switch-over
#pragma config FSOSCEN   = OFF           //Secondary Oscillator Enable
#pragma config CP        = OFF           //Code Protect
#pragma config BWP       = OFF           //Boot Flash Write Protect
#pragma config PWP       = OFF           //Program Flash Write Protect
#pragma config ICESEL    = ICS_PGx2      //ICE/ICD Comm Channel Select
//#pragma config DEBUG    = OFF            //Debugger Disabled for Starter Kit - ENABLE for RELEASE

//Application Defines
#define SYS_FREQ            (80000000)               //Operation Frequency
#define TOGGLES_PER_SEC     40000                    //Square Wave frequency for IR LED -Not needed anymore.
#define IR_TICK_RATE        (SYS_FREQ/2/TOGGLES_PER_SEC)
#define CORE_TICK_RATE      (SYS_FREQ/2/1000)        //Tick per millisecond for core timer
#define PRESCALE            256                      //Prescale for timer
#define TIMER_TICK_RATE     (SYS_FREQ/2/PRESCALE)    //Tick per second for timers
#define KEY_CLEAR           mPORTBSetBits(BIT_0 | BIT_1 | BIT_2 | BIT_3)
#define KEY_ROW_1           mPORTBClearBits(BIT_0);mPORTBSetBits(BIT_1 | BIT_2 | BIT_3)
#define KEY_ROW_2           mPORTBClearBits(BIT_1);mPORTBSetBits(BIT_0 | BIT_2 | BIT_3)
#define KEY_ROW_3           mPORTBClearBits(BIT_2);mPORTBSetBits(BIT_0 | BIT_1 | BIT_3)
#define KEY_ROW_4           mPORTBClearBits(BIT_3);mPORTBSetBits(BIT_0 | BIT_1 | BIT_2)
#define KEY_COL_1           PORTBbits.RB4
#define KEY_COL_2           PORTBbits.RB5
```

```c
#define KEY_COL_3           PORTBbits.RB13
#define KEY_COL_4           PORTBbits.RB9
#define IF_KEY_1            b[0]==(0x0001)      //If Keypad Key 1
#define IF_KEY_2            b[0]==(0x0002)      //If Keypad Key 2
#define IF_KEY_3            b[0]==(0x0004)      //If Keypad Key 3
#define IF_KEY_A            b[0]==(0x0008)      //If Keypad Key A
#define IF_KEY_4            b[0]==(0x0010)      //If Keypad Key 4
#define IF_KEY_5            b[0]==(0x0020)      //If Keypad Key 5
#define IF_KEY_6            b[0]==(0x0040)      //If Keypad Key 6
#define IF_KEY_B            b[0]==(0x0080)      //If Keypad Key B
#define IF_KEY_7            b[0]==(0x0100)      //If Keypad Key 7
#define IF_KEY_8            b[0]==(0x0200)      //If Keypad Key 8
#define IF_KEY_9            b[0]==(0x0400)      //If Keypad Key 9
#define IF_KEY_C            b[0]==(0x0800)      //If Keypad Key C
#define IF_KEY_STAR         b[0]==(0x1000)      //If Keypad Key *
#define IF_KEY_0            b[0]==(0x2000)      //If Keypad Key 0
#define IF_KEY_POUND        ((b[0]==(0x4000))|(b[0]==(0x4400))|(b[0]==(0x4404))|(b[0]==(0x4444))) //If Keypad Key #
#define IF_KEY_D            ((b[0]==(0x8000))|(b[0]==(0xC000))) //If Keypad Key D
#define KEY_1               (0x0001)            //Keypad Key 1
#define KEY_2               (0x0002)            //Keypad Key 2
#define KEY_3               (0x0004)            //Keypad Key 3
#define KEY_A               (0x0008)            //Keypad Key A
#define KEY_4               (0x0010)            //Keypad Key 4
#define KEY_5               (0x0020)            //Keypad Key 5
#define KEY_6               (0x0040)            //Keypad Key 6
#define KEY_B               (0x0080)            //Keypad Key B
#define KEY_7               (0x0100)            //Keypad Key 7
#define KEY_8               (0x0200)            //Keypad Key 8
#define KEY_9               (0x0400)            //Keypad Key 9
#define KEY_C               (0x0800)            //Keypad Key C
#define KEY_STAR            (0x1000)            //Keypad Key *
#define KEY_0               (0x2000)            //Keypad Key 0
#define KEY_POUND           (((0x4000))|((0x4400))|((0x4404))|((0x4444))) //Keypad Key #
#define KEY_D               (((0x8000))|((0xC000))) //Keypad Key D
#define CN_CONFIG           (CN_ON | CN_IDLE_CON)                   //Enable change notice on idle
#define CN_PIN_MOTOR        (CN11_ENABLE)                           //RG9 for motor tracking
#define CN_PIN_RESET        (CN10_ENABLE)                           //RG8 for motor reset
#define CN_CUP              (CN9_ENABLE)                            //RG7 for cup
#define CN_PULLUP_MOTOR     (CN11_PULLUP_ENABLE)                    //330K pullup resistor
#define CN_PULLUP_RESET     (CN10_PULLUP_ENABLE)                    //330K pullup resistor
#define CN_PULLUP_CUP       (CN9_PULLUP_ENABLE)                     //330K pullup resistor
#define CN_INTERRUPT        (CHANGE_INT_ON | CHANGE_INT_PRI_4)      //Reset Priority 4
#define DEST_1              96                                      //Post for Cup 1
#define DEST_2              263                                     //Post for Cup 2
#define DEST_3              428                                     //Post for Cup 3
#define DEST_4              594                                     //Post for Cup 4
#define DEST_5              761                                     //Post for Cup 5
#define DEST_6              928                                     //Post for Cup 6
#define IF_MAIN             (state==0)                              //State = main menu
```

```
#define IF_CONFI          (state==1)                        //State = configuration menu
#define IF_EXECUTING      (state==2)                        //State = executing order
#define IF_ST_A           (state==3)                        //State = Valve A
#define IF_ST_B           (state==4)                        //State = Valve B
#define IF_ST_C           (state==5)                        //State = Valve C
#define MAIN              0                                 //Main state code
#define CONFI             1                                 //Configuration state code
#define EXECUTING         2                                 //Execution state code
#define ST_A              3                                 //Valve A state code
#define ST_B              4                                 //Valve B state code
#define ST_C              5                                 //Valve C state codes
#define VALVE_A_ON        mPORTDSetBits(BIT_0)              //Enable Vavle A
#define VALVE_B_ON        mPORTDSetBits(BIT_1)              //Enable Vavle B
#define VALVE_C_ON        mPORTDSetBits(BIT_2)              //Enable Vavle C
#define VALVES_OFF        mPORTDClearBits(BIT_0|BIT_1|BIT_2) //Disable All Vavle
#define TABLE_ON          mPORTDSetBits(BIT_3)              //Rotate Table
#define TABLE_OFF         mPORTDClearBits(BIT_3)            //Stop Table
#define IF_CUP_SIGNAL     (!PORTGbits.RG7)                  //IR Detector Signal
#define CUP_STOP          mPORTFClearBits(BIT_0)            //Stop cup dispensing
#define CUP_START         mPORTFSetBits(BIT_0)              //Start cup dispensing

//Prototypes
void DelayMs(unsigned int);
void InitIREmitter(void);
void InitCupTable(void);
void InitKeys(void);
void InitPouring(void);
int  readKEY(void);
void getK(volatile int *c);

//Global Vars
volatile short int pos=0;                          //position of turn table
volatile short int dest=DEST_1;                    //Init cup destination to first spot
volatile short int cup=0;                          //cup positioned
volatile short int state=0;                        //menu state: 0-main, 1-config, 2-executing order, 3-A, 4-B, 5-C
volatile short int calib=0;                        //table calibiration
volatile unsigned short int valve_times[3]={6000,6000,6000}; //valves' timing settings
volatile short int order[3]={0,0,0};               //beer order goes here

// BeerBot main code
int main(void){
    volatile int b[2];

    //Configure the device for maximum performance, but do not change the PBDIV clock divisor.
    //Given the options, this function will change the program Flash wait states,
    //RAM wait state and enable prefetch cache, but will not change the PBDIV.
    //The PBDIV value is already set via the pragma FPBDIV option above.
    SYSTEMConfig(SYS_FREQ, SYS_CFG_WAIT_STATES | SYS_CFG_PCACHE);
    DBINIT();                                       //GET RID OF THIS AFTER TESTING IS DONE
```

```c
    DBPRINTF("BeerBot: can't wait to make you drunk.\n");         //GET RID OF THIS AFTER TESTING IS DONE

    //Initialize System Environment
    //InitIREmitter();
    InitKeys();
    InitPouring();
    InitCupTable();                                               //Call this function last!

    //Enable Systemwide Mutivecotr Interrupts
    INTEnableSystemMultiVectoredInt();
    mPORTDSetPinsDigitalOut(BIT_0);
    mPORTDSetPinsDigitalIn(BIT_1);

    while(1){ //Never exit this loop!
            DBPRINTF("State = %u\n",state);
        if (IF_MAIN){ //Main Menu
            getK(b);
            if (IF_KEY_1) {state=ST_A;}                           //select Valve A
            if (IF_KEY_2) {state=ST_B;}                           //select Valve B
            if (IF_KEY_3) {state=ST_C;}                           //select Valve C
            if (IF_KEY_B){                                        //reset order
                DBPRINTF("RESET ORDER \n");
                state=MAIN;
                order[0]=0;
                order[1]=0;
                order[2]=0;
            }
            if (IF_KEY_A){                                        //execute order
                if (order[0]+order[1]+order[2]>0){                //check that order is entered
                    state=EXECUTING;                              //State = Executing Order
                    switch (dest){                                //Go to next cup spot
                        case DEST_1 : dest=DEST_2;
                                      break;
                        case DEST_2 : dest=DEST_3;
                                      break;
                        case DEST_3 : dest=DEST_4;
                                      break;
                        case DEST_4 : dest=DEST_5;
                                      break;
                        case DEST_5 : dest=DEST_6;
                                      break;
                        case DEST_6 : dest=DEST_1;
                                      break;
                    }
                    TABLE_ON;                                     //Rotate Table
                }
            }
            if (IF_KEY_D) {state=CONFI;DBPRINTF("CONFIG\n");}                //select configuration menu
```

```c
        } //MAIN
        else {
        if (IF_ST_A){ //Valve A
            getK(b);
            if (IF_KEY_B){                                        //reset order
                state=MAIN;
                order[0]=0;
                order[1]=0;
                order[2]=0;
            }
            if (IF_KEY_4)
                if (order[0]+order[1]+order[2]<6){
                state=MAIN;                                       //Return to main menu
                order[0]=1;
                }
            else state=MAIN;                                      //Return to main menu if cant order any more
            if (IF_KEY_5)
                if (order[0]+order[1]+order[2]<5){
                state=MAIN;                                       //Return to main menu
                order[0]=2;
                }
            else state=MAIN;                                      //Return to main menu if cant order any more
            if (IF_KEY_6)
                if (order[0]+order[1]+order[2]<4){
                state=MAIN;                                       //Return to main menu
                order[0]=3;
                }
            else state=MAIN;                                      //Return to main menu if cant order any more
            if (IF_KEY_7)
                if (order[0]+order[1]+order[2]<3){
                state=MAIN;                                       //Return to main menu
                order[0]=4;
                }
            else state=MAIN;                                      //Return to main menu if cant order any more
            if (IF_KEY_8)
                if (order[0]+order[1]+order[2]<2){
                state=MAIN;                                       //Return to main menu
                order[0]=5;
                }
            else state=MAIN;                                      //Return to main menu if cant order any more
            if (IF_KEY_9)
                if (order[0]+order[1]+order[2]<1){
                state=MAIN;                                       //Return to main menu
                order[0]=6;
                }
            else state=MAIN;                                      //Return to main menu if cant order any more
        } //ST_A
        else{
        if (IF_ST_B){ //Valve B
```

```c
        getK(b);
        if (IF_KEY_B){                                          //reset order
            state=MAIN;
            order[0]=0;
            order[1]=0;
            order[2]=0;
        }
        if (IF_KEY_4)
            if (order[0]+order[1]+order[2]<6){
            state=MAIN;                                         //Return to main menu
            order[1]=1;
            }
        else state=MAIN;                                        //Return to main menu if cant order any more
        if (IF_KEY_5)
            if (order[0]+order[1]+order[2]<5){
            state=MAIN;                                         //Return to main menu
            order[1]=2;
            }
        else state=MAIN;                                        //Return to main menu if cant order any more
        if (IF_KEY_6)
            if (order[0]+order[1]+order[2]<4){
            state=MAIN;                                         //Return to main menu
            order[1]=3;
            }
        else state=MAIN;                                        //Return to main menu if cant order any more
        if (IF_KEY_7)
            if (order[0]+order[1]+order[2]<3){
            state=MAIN;                                         //Return to main menu
            order[1]=4;
            }
        else state=MAIN;                                        //Return to main menu if cant order any more
        if (IF_KEY_8)
            if (order[0]+order[1]+order[2]<2){
            state=MAIN;                                         //Return to main menu
            order[1]=5;
            }
        else state=MAIN;                                        //Return to main menu if cant order any more
        if (IF_KEY_9)
            if (order[0]+order[1]+order[2]<1){
            state=MAIN;                                         //Return to main menu
            order[1]=6;
            }
        else state=MAIN;                                        //Return to main menu if cant order any more
    } //ST_B
    else{
    if (IF_ST_C){ //Valve C
        getK(b);
        if (IF_KEY_B){                                          //reset order
            state=MAIN;
```

```
                order[0]=0;
                order[1]=0;
                order[2]=0;
            }
            if (IF_KEY_4)
                if (order[0]+order[1]+order[2]<6){
                state=MAIN;                                     //Return to main menu
                order[2]=1;
                }
            else state=MAIN;                                    //Return to main menu if cant order any more
            if (IF_KEY_5)
                if (order[0]+order[1]+order[2]<5){
                state=MAIN;                                     //Return to main menu
                order[2]=2;
                }
            else state=MAIN;                                    //Return to main menu if cant order any more
            if (IF_KEY_6)
                if (order[0]-order[1]+order[2]<4){
                state=MAIN;                                     //Return to main menu
                order[2]=3;
                }
            else state=MAIN;                                    //Return to main menu if cant order any more
            if (IF_KEY_7)
                if (order[0]-order[1]+order[2]<3){
                state=MAIN;                                     //Return to main menu
                order[2]=4;
                }
            else state=MAIN;                                    //Return to main menu if cant order any more
            if (IF_KEY_8)
                if (order[0]-order[1]+order[2]<2){
                state=MAIN;                                     //Return to main menu
                order[2]=5;
                }
            else state=MAIN;                                    //Return to main menu if cant order any more
            if (IF_KEY_9)
                if (order[0]+order[1]+order[2]<1){
                state=MAIN;                                     //Return to main menu
                order[2]=6;
                }
            else state=MAIN;                                    //Return to main menu if cant order any more
        } //ST_C
        else{
        if (IF_CONFI){ //Configuration Menu
            DBPRINTF("Config Loc\n");
            getK(b);
            if (IF_KEY_STAR) valve_times[0]=CORE_TICK_RATE*(b[1]-50);   //get new time for valve A
            if (IF_KEY_0) valve_times[1]=CORE_TICK_RATE*(b[1]-50);      //get new time for valve B
            if (IF_KEY_POUND) valve_times[2]=CORE_TICK_RATE*(b[1]-50);  //get new time for valve C
            if (IF_KEY_C) state=MAIN;                                   //return to main menu
```

```
        } //CONFI
        } //ST_C else
        } //ST_B else
        } //ST_A else
        } //Main else
//  getK(b);
            if (IF_KEY_0) DBPRINTF("Key = 0 ");
            if (IF_KEY_1) DBPRINTF("Key = 1 ");
            if (IF_KEY_2) DBPRINTF("Key = 2 ");
            if (IF_KEY_3) DBPRINTF("Key = 3 ");
            if (IF_KEY_4) DBPRINTF("Key = 4 ");
            if (IF_KEY_5) DBPRINTF("Key = 5 ");
            if (IF_KEY_6) DBPRINTF("Key = 6 ");
            if (IF_KEY_7) DBPRINTF("Key = 7 ");
            if (IF_KEY_8) DBPRINTF("Key = 8 ");
            if (IF_KEY_9) DBPRINTF("Key = 9 ");
            if (IF_KEY_A) DBPRINTF("Key = A ");
            if (IF_KEY_B) DBPRINTF("Key = B ");
            if (IF_KEY_C) DBPRINTF("Key = C ");
            if (IF_KEY_D) DBPRINTF("Key = D ");
            if (IF_KEY_STAR) DBPRINTF("Key = * ");
            if (IF_KEY_POUND) DBPRINTF("Key = # ");
            DBPRINTF("Code = %X | Time = %ums \n", b[0],b[1]);


} //Main

/****************************************************************************
*   DelayMs(mSec)
*   ----------------------
*   This functions provides a software millisecond delay
****************************************************************************/
void DelayMs(unsigned int msec){
    unsigned int tWait, tStart;

    tWait=(SYS_FREQ/2000)*msec;
    tStart=ReadCoreTimer();
    while((ReadCoreTimer()-tStart)<tWait);      //wait for the time to pass
} //DelayMs

/****************************************************************************
*   InitIREmitter() - Not Used anymore
*   ----------------------
*   This functions enables OC1 with 50% duty cycle and IR_TICK_RATE period
****************************************************************************/
void InitIREmitter(void){
    //Enable Timer2 | ON on idle | Int Priority = 5 | Sub Priority = 0 | Prescaler 1:1 , Count
    OpenTimer2(T2_ON | T2_IDLE_CON | T2_INT_PRIOR_5 | T2_INT_SUB_PRIOR_0 | T2_PS_1_1, IR_TICK_RATE);
    //Enable OC1 | 16 bit Mode  | Timer2 is selected | Continuous O/P   | OC Pin High , S Compare value, Compare value
```

```c
    OpenOC4(OC_ON | OC_TIMER_MODE16 | OC_TIMER2_SRC | OC_CONTINUE_PULSE | OC_LOW_HIGH , 0, 0);
} //InitIREmitter

/****************************************************************************
*   InitCupTable()
*   ----------------------
*   This functions initializes Turntable I/O pins and interrupts.
****************************************************************************/
void InitCupTable(void){
    unsigned int temp;                                          //temp variable

    PORTSetPinsDigitalIn(IOPORT_G, BIT_7 | BIT_8 | BIT_9);      //RG7, RG8 and RG9 are Digital inputs
    PORTSetPinsDigitalOut(IOPORT_D, BIT_3);                     //RD3 is Digital output - TTable
    PORTSetPinsDigitalOut(IOPORT_F, BIT_0);                     //RF0 is Digital output - Cup Dispenser
    mPORTDClearBits(BIT_3);                                     //Stop table
    mPORTFClearBits(BIT_0);                                     //Stop cup dispenser
    mCNOpen(CN_CONFIG, CN_CUP | CN_PIN_RESET | CN_PIN_MOTOR, CN_PULLUP_CUP | CN_PULLUP_RESET | CN_PULLUP_MOTOR);
//Setup CN for Motor and Cup signals.
    temp = mPORTGReadBits(BIT_7 | BIT_8 | BIT_9);               //read port(s) to clear mismatch on CN pins
    ConfigIntCN(CN_INTERRUPT);                                  //Motor reset now has interrupt
    calib=1;                                                    //Turn Table driver will calibirate table now
} //InitTable

/****************************************************************************
*   InitKeys()
*   ----------------------
*   This functions initializes I/O pins for Keypad
****************************************************************************/
void InitKeys(void){
    mPORTBSetPinsDigitalOut(BIT_0 | BIT_1 | BIT_2 | BIT_3);     //configure RB0-RB3 as output
    mPORTBSetPinsDigitalIn(BIT_4 | BIT_5 | BIT_13 | BIT_9);     //configure RB4,RB5,RB8,RB9 as input
    mPORTBSetBits(BIT_0 | BIT_1 | BIT_2 | BIT_3);               //initialize these pins HI
} //InitKeys

/****************************************************************************
*   InitPouring()
*   ----------------------
*   This functions initializes I/O pins for Pouring
****************************************************************************/
void InitPouring(void){
    mPORTDSetPinsDigitalOut(BIT_0 | BIT_1 | BIT_2);             //configure PORTD
    mPORTDClearBits(BIT_0 | BIT_1 | BIT_2);                     //initialize these pins LOW
} //InitPouring

/****************************************************************************
*   readKEY()
*   ----------------------
*   This functions returns integer that has encoded info for keys pressed
****************************************************************************/
```

```c
int readKEY(void){   //returns 0..F if keys pressed, 0 = none
    int c = 0;               //clear input
    int temp = 0;            //temp var for port read.

    KEY_ROW_1;
    temp = mPORTBRead();//DelayMs(1);
    if (!KEY_COL_1) // KEY 1
        c  = 0b0000000000000001;
    if (!KEY_COL_2) // KEY 2
        c  = 0b0000000000000010;
    if (!KEY_COL_3) // KEY 3
        c  = 0b0000000000000100;
    if (!KEY_COL_4) // KEY A
        c  = 0b0000000000001000;

    KEY_ROW_2;
    temp = mPORTBRead();//DelayMs(1);
    if (!KEY_COL_1) // KEY 4
        c  = 0b0000000000010000;
    if (!KEY_COL_2) // KEY 5
        c  = 0b0000000000100000;
    if (!KEY_COL_3) // KEY 6
        c  = 0b0000000001000000;
    if (!KEY_COL_4) // KEY B
        c  = 0b0000000010000000;

    KEY_ROW_3;
    temp = mPORTBRead();//DelayMs(1);
    if (!KEY_COL_1) // KEY 7
        c  = 0b0000000100000000;
    if (!KEY_COL_2) // KEY 8
        c |= 0b0000001000000000;
    if (!KEY_COL_3) // KEY 9
        c |= 0b0000010000000000;
    if (!KEY_COL_4) // KEY C
        c |= 0b0000100000000000;

    KEY_ROW_4;
    temp = mPORTBRead();//DelayMs(2);
    if (!KEY_COL_1) // KEY *
        c |= 0b0001000000000000;
    if (!KEY_COL_2) // KEY 0
        c |= 0b0010000000000000;
    if (!KEY_COL_3) // KEY #
        c |= 0b0100000000000000;
    if (!KEY_COL_4) // KEY D
        c  = 0b1000000000000000;
    return c;
} // readK
```

```c

/*****************************************************************************
*   getK(int c[2])
*   ----------------------
*   This functions provides debounce support for keypad.
*****************************************************************************/
void getK(volatile int *c){ //wait for a key pressed and debounce
    int i=0, r=0, j=0;

// 1. wait for a key pressed for at least .1sec
    do{
        DelayMs(10);
        if ((c[0] = readKEY())){
            if (c[0]!=r)                    //if more than one button pressed
                r = c[0];                   //take the new code
            i++;
        }
        else i=0;
    } while (i<5);

// 2. wait for key released for at least .1 sec
    i =0;
    if (IF_CONFI){                          //turn valve ON if we are in configuration
        switch (r){
            case KEY_STAR : mPORTDSetBits(BIT_0);
                            DBPRINTF("Valve A on\n");
                            break;
            case KEY_0    : mPORTDSetBits(BIT_1);
                            DBPRINTF("Valve B on\n");
                            break;
            case KEY_POUND: mPORTDSetBits(BIT_2);
                            break;
        }
    }
    do {
        DelayMs(10);
        if ((c[0] = readKEY())){
            if (c[0]!=r)                     //if more then one button pressed
                r = c[0];                    //take the new code
            i=0;
            j++; // keep counting
        }
        else i++;
    } while (i<5);
    if (IF_CONFI) {mPORTDClearBits(BIT_0 | BIT_1 | BIT_2);
    DBPRINTF("VALVES OFF\n");}
// 3. return lenght of key being pressed in ms
    c[1]=(j*10-100);                         //it takes 10ms for j to increment and 100ms to exit both loops

```

```
// 4. return key code
    c[0] = r;

} // getK

/*****************************************************************************
*   _ISR - Priority 4
*   ----------------------
*   ISR for Motor Tracking and Reset signals.
*****************************************************************************/
void __ISR(_CHANGE_NOTICE_VECTOR, ipl4) ChangeNotice_Reset(void)
{
    unsigned int temp;

    temp = mPORTGReadBits(BIT_7 | BIT_8 | BIT_9);                  //read port(s) to clear mismatch on CN pins
    mCNClearIntFlag();                                             //clear the interrupt flag
    if (IF_CUP_SIGNAL) CUP_STOP;                                   //stop cup dispensing if ISR started by IR
Detector
    else{ //Check for calibration
        if (calib){ //Calibirate Table
            if (mPORTGReadBits(BIT_8)){ //RESET Signal
                pos=0;                                             //Woot. We know we are at POS=0
                calib=0;                                           //Yay. Table is calibirated.
                mPORTGClearBits(BIT_0);                            //Hey. Stop the table.
            }
        }

        else{ //Normal operation
            if (mPORTGReadBits(BIT_9)){ //TRACKING Signal
                pos--;                                             //Increment position
                if (pos==dest){                                    //Check if we reached the destination
                    TABLE_OFF;                                     //Stop table
                    if (!IF_CUP_SIGNAL) CUP_START;                 //Drop cup if spot is empty
                    if (order[0]){                                 //Check if beer A is requested
                        order[0]--;                                //Decrement by 1
                        VALVE_A_ON;                                //Start pouring beer A
                        UpdateCoreTimer(valve_times[0]);           //Set alarm. Zzz.
                        mCTClearIntFlag();                         //Enable alarm.
                    }

                    else{ if (order[1]){                           //Check if beer A is requested
                            order[1]--;                            //Decrement by 1
                            VALVE_B_ON;                            //Start pouring beer A
                            UpdateCoreTimer(valve_times[1]);       //Set alarm. Zzz.
                            mCTClearIntFlag();                     //Enable alarm.
                        }

                        else{ if (order[2]){                       //Check if beer A is requested
                                order[2]--;                        //Decrement by 1
                                VALVE_C_ON;                        //Start pouring beer A
                                UpdateCoreTimer(valve_times[2]);   //Set alarm. Zzz.
                                mCTClearIntFlag();                 //Enable alarm.
```

```
                        }
                        else state=MAIN;                                   //No Orders - go to main
                    }
                }
            } //pos==dest
        } //Tracking signal
        if (mPORTGReadBits(BIT_8)){ //RESET Signal
            if (pos>900) pos=0;                                            //Position is 0. Simple!
        }
    }
  }
} //ISR for TurnTable

/************************************************************************
*   _ISR - Priority 5
*   ----------------------
*   ISR for Pouring subsystem
************************************************************************/
void __ISR(_CORE_TIMER_VECTOR, ipl5) CoreTimerHandler(void)
{
    VALVES_OFF;                                                //Stop Pouring on all valves
    if (order[0]+order[1]+order[2]==0) state=MAIN;             //Pouring done - returning to main menu
                                        //
} //ISR for pouring system
```

Appendix E: Electrical Equipment

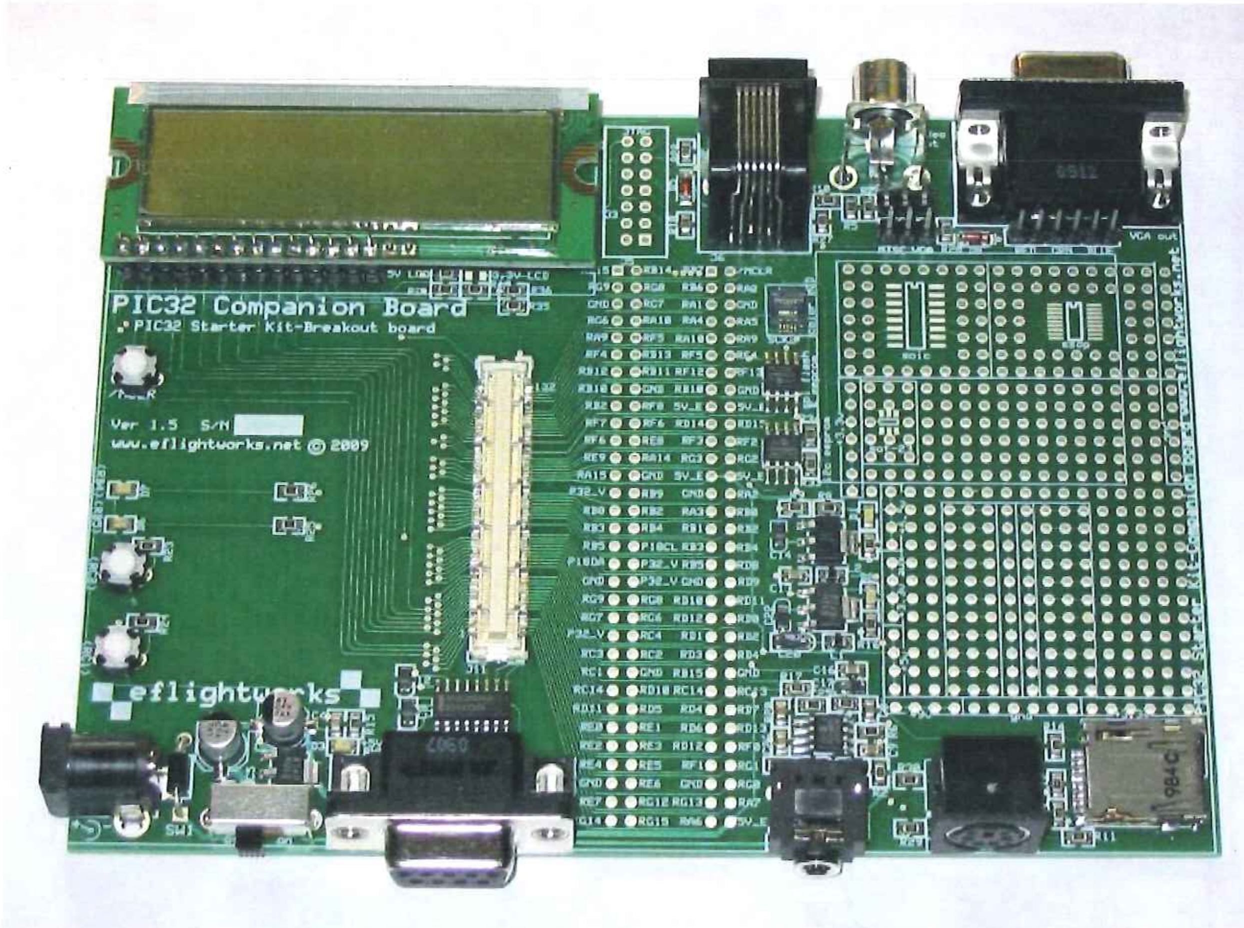


Figure F-1: Eflightworks.net PIC32 I/O Expansion Board v1.5

Figure F-2: 5 VDC Voltage Regulator LM340T-5.0-ND

Figure F-3: Solenoid valve 72R9DGV-12VDC

Figure F-4: Power Supply Altech #PS-S6012

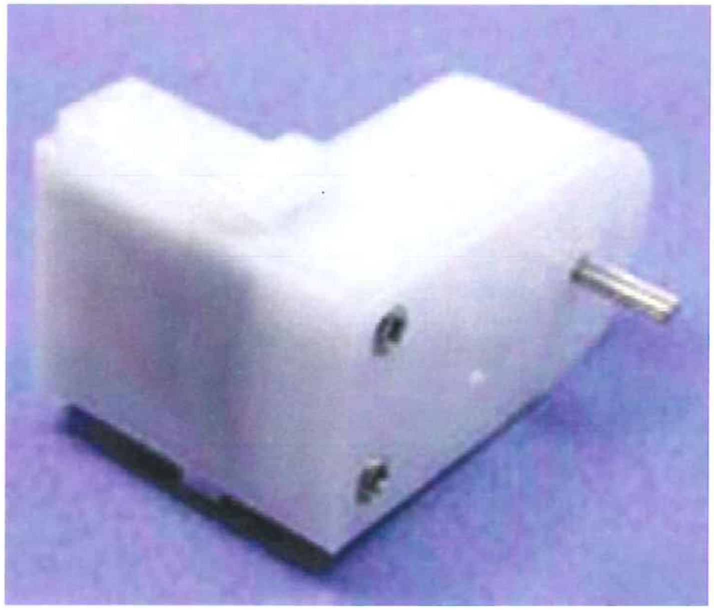


Figure F-5: DC Motor with Gears [1].

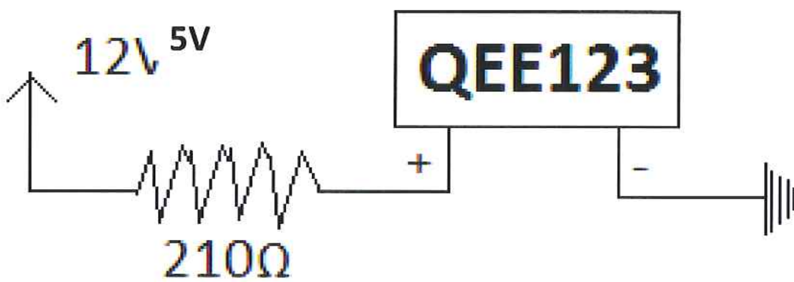


Figure F-6: IR Emitter Circuit.

[1] Model number is unknown due to team member dropping out without supplying this information.

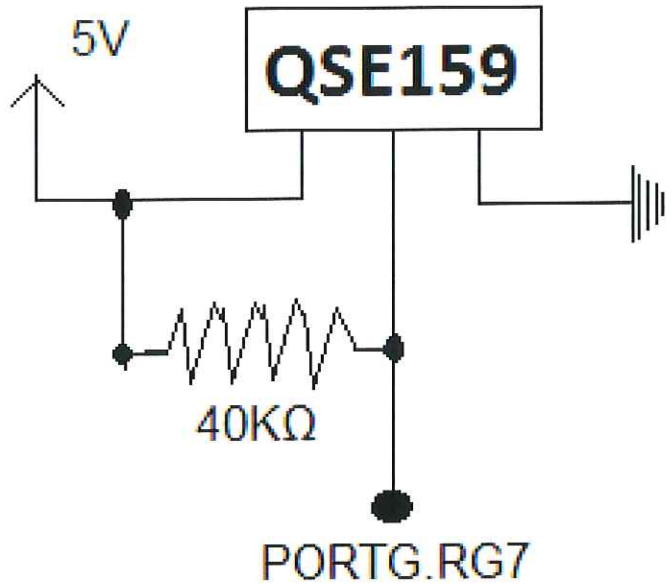


Figure F-7: IR Detector Circuit

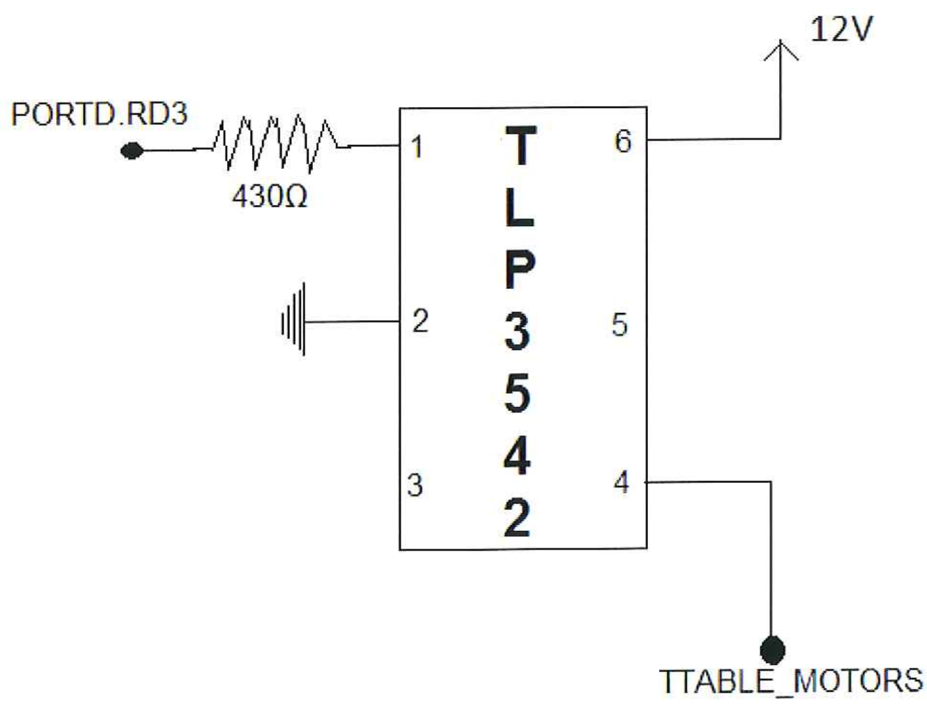


Figure F-8: Turntable Motor Control Circuit

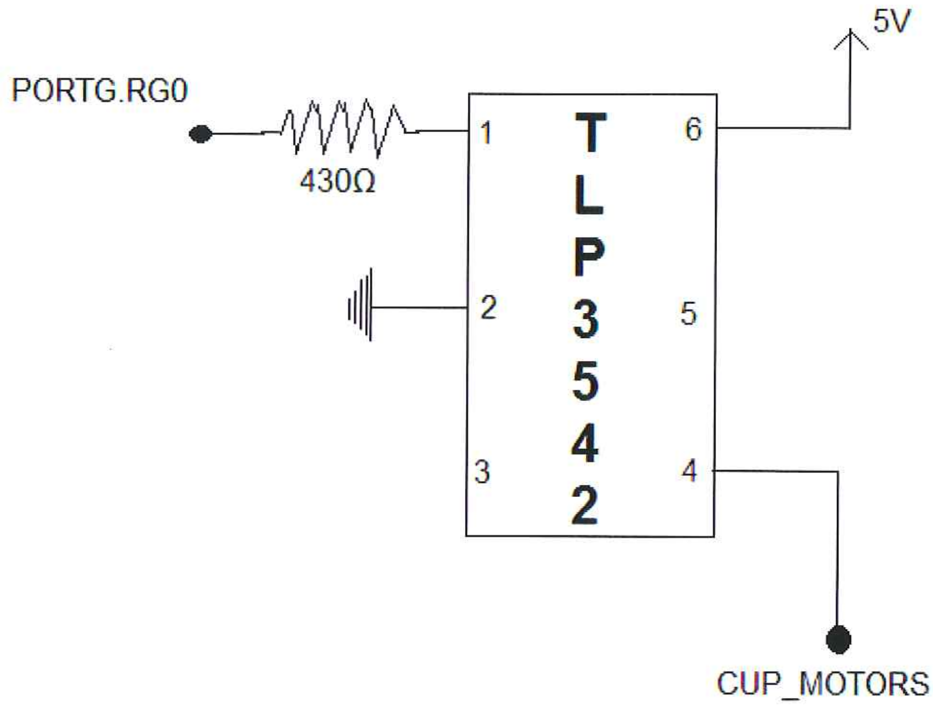


Figure F-9: Turntable Motor Control Circuit

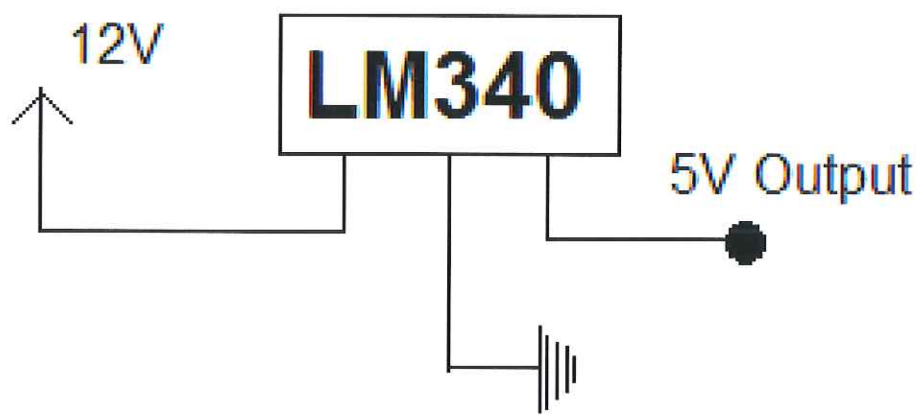


Figure F-10: 5VDC Regulator Circuit

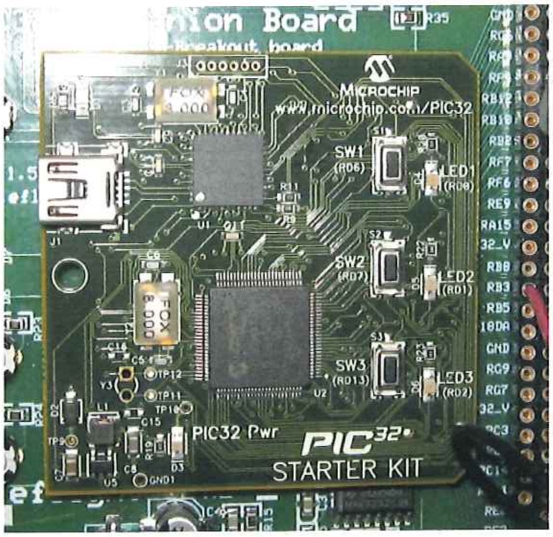


Figure F-11: PIC32mx Starter Kit (Plugged into the expansion board)

Figure F-12: Keypad Unit with DB25 Connector

Figure F-13: IR Emitter and Detector

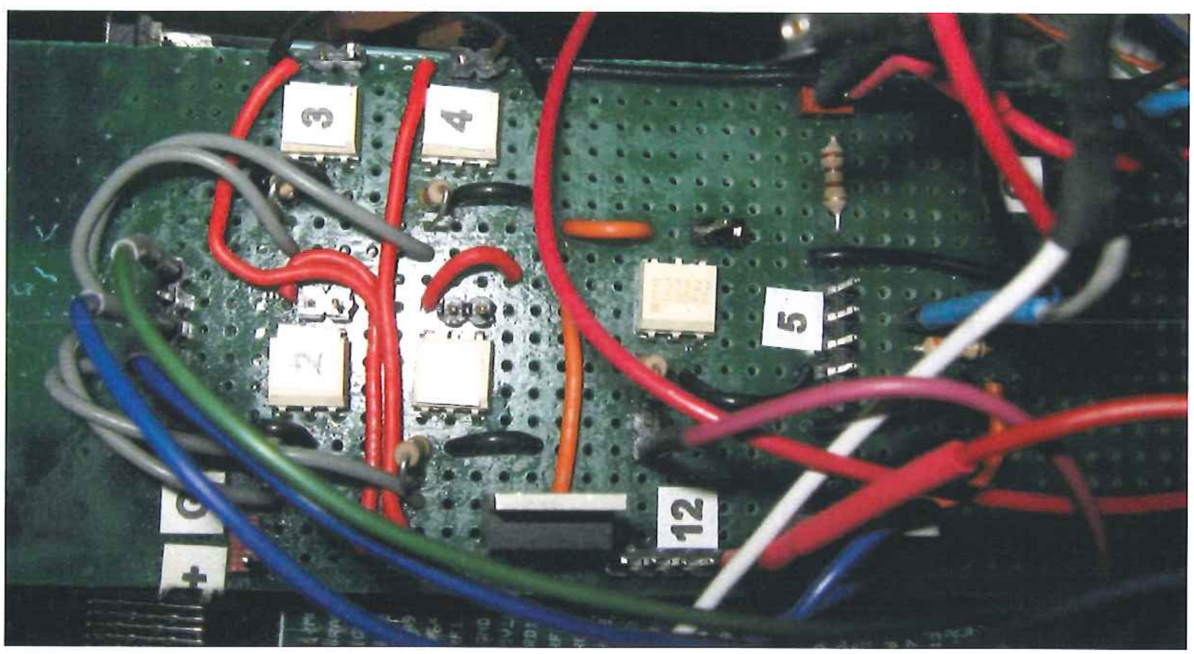


Figure F-13: Custom Circuit for Controlling Valves, Turntable, Cup Dispenser.