Design of an Automated Drink Dispenser with Mobile Application (Polytéleia)

Group members

Matthew Whitlow, email: mw83@students.uwf.edu

Taylor Lookabaugh, email: tjl13@students.uwf.edu

Stefan Wheeless, email: sbw20@students.uwf.edu

Annaly Benitez, e-mail: ab111@students.uwf.edu

Team Leader

Annaly Benitez

Speciality Advisor

Dr. A. Fuchs

Course Instructor and Advisor

Dr. T Gilbar

Capstone Design I

EGN 4952L

Spring 2018

We certify that this assignment is the result of our own efforts.

Group Members	E-mail	Camps	Specilaity Advisor	Project
Matthew Whitlow	mw83@students.uwf.edu	UWF Emerald Coast	Dr. A. Fuchs	Design of an Automated Drink Dispenser with Mobile Application
Taylor Lookabaugh	tjl13@students.uwf.edu			
Stefan Wheeless	sbw20@students.uwf.edu			
Annaly Benitez	ab111@students.uwf.edu			

Abstract:

This project was designed to provide a quick precise and convenient solution for dispensing mixed beverages for commercial and at-home applications. The automated drink dispenser solves the issue of over pouring and profit loss by inexperienced, or overzealous bartenders. It can save a business money by being an “extra hand” during busy business hours instead of hiring a bar back. For at home use, the device is a fun luxury entertainment system that can help the host take care of guests.

Table of Contents

Table of Figures

Table of Tables

1. Introduction:

The U.S. beverage market is a $354.2 billion industry with alcoholic beverages making 60% of the revenues [1]. Spirits are consumed regularly on all occasions, locations, and times. The necessity of skilled and knowledgeable bartenders is constant because of this. Unfortunately, businesses lose thousands of dollars monthly because of over pouring by their employees. The solution is the creation of an automated drink dispenser. With the system's precision and accuracy, beverages can be produced almost perfectly without virtually any over pour. The device is so easy to use that at home application is available. The drink menu is easily accessed using a password on the LCD touch screen, or by scanning a driver license ID on the mobile application. It consists of various drink recipes and once one is selected, the unit goes from stationary mode to preparation mode. Within a few seconds, the drink dispenser successfully produces a delicious beverage of your choice.

2. Problem Definition:

2.1 Problem/Need:

The intent of this device would be to provide the user with a quick, precise, and convenient solution for dispensing mixed beverages for commercial and at-home applications. In a commercial setting, the device could be used in restaurants to eliminate over-pour profit loss, which can result in thousands of dollars lost per month. Additionally, it can save a business money by being an "extra hand" during busy business hours instead of hiring a bar back. For at home use, the device is a fun luxury entertainment system that can help the host take care of guests.

2.2 Intended user(s) and use(s):

The intended users for this automatic drink maker would be restricted to people 21 years of age and above while an alcoholic beverage is being ordered. For commercial use the employee wouldn't have to be of age depending on the state the device is being used in. Otherwise, nonalcoholic beverages can be dispensed to minors.

2.3 Assumptions and limitations:

A major assumption would be that the user is a responsible adult of at least 21 years of age that wouldn't give access to a minor. Additionally, during presentations the device would be dispensing fake cocktail recipes instead of liquor based recipes in compliance with university policy. Another limitation would be the number of beverages that the device is capable of making successfully, as we will be limited in the number of liquor/mixers available to the device due to space.

2.4 Design objectives:

The design objective is to have a standalone automatic drink maker that successfully delivers a beverage that was selected by the user. The device will be easily disassembled for easy

storage, transportation, and cleaning. The user will have access to the drink menu once the user's age is verified through the ID reader programmed into the mobile application. They would also have access to the menu at the unit itself, using a password which will have to be inputted on a LCD touch screen. The mobile application will only have the ability to display the menu and order drinks, while the unit will display a settings menu as well. The settings menu will allow the user to make modifications to update password settings, and user profiles. The main menu will display drinks listed alphabetically. Once the user makes their selection the device will be signaled to start. A combination of three LED colors will be used to indicated which process is being executed (stationary, preparation, dispensing, completion). During the preparation stage, the glass will glide back and forth to different locations based on the recipe selected and receive liquids from the dispensaries until the recipe has been completed. Finally, the cup will return to the stationary position and the LED light will change to indicate that the drink recipe has been accurately completed.

3. Design Specifications:

1. The structure of the design will be made from wood/metal parts.
2. The units main controller will be accessed using a LCD touchscreen device that will be password protected. There will be three menus: a settings menu, kid's menu, and drink menu. The settings menu will give the user multiple options such as: password reset, and user profile settings.
3. The mobile application will be developed using Android app development. The app will have Bluetooth and Wi-Fi capabilities to be able to access the device inside/outside the home. It will display a drink menu just like the main controller, but will be accessed by scanning the 2D barcode on an ID.
4. The Raspberry Pi 3 will be used for communication between the GUI, mobile application and the Arduino Mega which is used for several controls on the device.
    - The cup will be moved using a stepper motor that will move the cup back and forth depending on the recipe.
    - Additionally, the liquid that will be held in upside down bottles will be dispensed using a water solenoid valve that will be programmed using a timer that continues to dispense liquid until the recipe amount is met, while right-side up containers will use an air pump for dispensing.
    - The changing LEDs based on the 3 modes will also be controlled with the Arduino Mega.

4. End-Product Description:

Introducing the newest development in smart home devices. This app activated, autonomous bartender provides the precision of a mixologist and convenience for at home entertainment! The autonomous bartender can be customized to be specific to one

user or expanded to fulfill many. Powered by Bluetooth and Wi-Fi capable devices the automatic drink maker leaves the power in your hands.

5. Design constraints:

This subsection species the major constraints that were considered during the design and implementation of the project.

5.1 Economical

Economic constraints will clearly be the cost to design and build the device. Searching for the most reliable products for the best price will be imperative. Materials and supplies necessary will have to be budgeted to successfully complete and make it affordable for the user.

5.2 Environmental

Environmental constraints could appear during testing and building. We would test different recipes using liquids like water to ensure our measurements and designs are accurate. We would want to reuse the water repeatedly. Also, any other material in the structure would need to be used and not wasted.

5.3 Social and Political

Social and political constraints would include safeguarding our device. The product is intended to produce mainly alcoholic beverages; therefore, we want to make sure that anyone that is not 21 years of age does not have access to the system. There will be an option for non-alcoholic beverages, but this will not give access to the adult beverage menu. The user should feel comfortable knowing that the alcoholic beverages are not easily accessible.

5.4 Ethical

Ethical constraints would include verifying the age of the user. Again, we want to make sure that the people that have access to the device are allowed to by law.

5.5 Health and Safety

Health and safety constraints would include water proofing all circuitry in the device. The system does have moving parts which have the potential of being hazardous to the user if not used with care. The machine must maintain sanitary conditions at all times as consumables are being produced. Therefore, the device must be cleaned carefully and often to maintain these conditions. Constructing the device out of material that can be easily cleaned would be necessary. Additionally, we would want to advocate responsible consumption of beverages.

5.6 Manufacturability and Sustainability

Manufacturability and sustainability constraints would include making sure the structure is strong and sound so that it lasts. Also, we would want to maintain a clean environment to avoid rust, or other debris that could cause problems in the future.

6. Engineering Standards:

ISO-13849-1 – Safety of Machinery -- Safety-Related Parts of Control Systems, General Principles for design

ISO12100 – Safety of Machinery – General Principles for design – Risk assessment and risk reduction

FDA Food Code 2009: Chapter 4 – Equipment, Utensils & Linens

ISO/IEC 17799 -- Code of practice for information security management

NFPA 70 -- GFCI

7. Technical Approach:

Polytéleia's power is supplied from a common 120V, 60Hz AC power main, connected via a NEMA 5-15P standard outlet. It will then be converted down to a useable 12V DC by an off-the-shelf DC power supply, which will provide power to the motor controllers, and the main system controller (the system controller will then redistribute power to the remaining, less power-hungry components). Functionally, the system will idle on the main drink selection screen until a drink is selected; either by the user choosing from the menu via the onboard interface (LCD screen and buttons), or by receiving an order via the mobile application using Bluetooth or Wi-Fi. If age verification methods are enabled and the selected drink is listed as being alcoholic, the system will await user age verification by the pre-set method – either a secret code entered via the menu navigation buttons or via an ID scan on the mobile app (system will return to initial state should verification fail). Once any necessary verifications are complete, the main control system will follow the menu selection commands whether it be the user settings path or a drink menu selection. If a listed recipe is selected, the main control system initiates by issuing commands to the conveyance stepper motor, air pumps and relevant solenoid actuators' controllers. The bottle dispenser consists of a normally closed valve and an electromagnetic solenoid. When a signal from the microcontroller energizes the solenoid, the value opens. Similarly, mixer dispensers consist of clear hoses driven by peristaltic air pumps. The amount of liquid dispensed will be a function of time. To further test this theory a measuring cup will be utilized while recording the solenoids energized time. The correct time will then be coded into the program for accurate mixing volumes. Once the dispenser operation is functional, the cup position can be implemented. The distance of the dispenser's relative to the conveyor is known therefore precise step signals can be sent to the stepper motor to ensure the cup on the conveyor is positioned under the dispenser.

7.1 Functional Block diagram

Standard wall power will be converted to 12V DC to provide power, while signaling will be achieved by standard 3.3V digital communications.

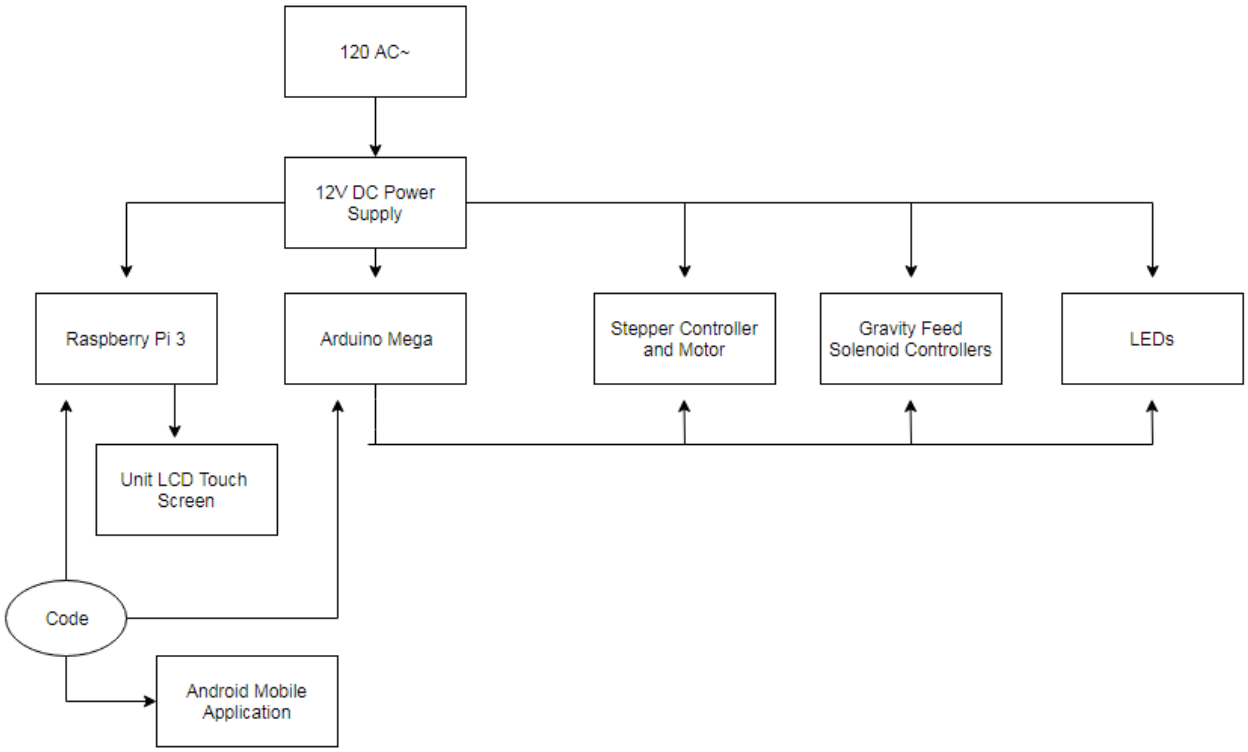

Figure 1. Polytéleia block diagram.

7.2 Functional Requirements for Blocks

The functional requirements of the blocks in Figure 1 are listed in Tables 1 through 8.

Table 1. Functional Requirements and Specifications of 12V DC Power Supply Block.

Requirements	Specifications
Functional Requirements	Rectify, filter, and regulate wall AC to steady DC
Performance Requirements	Input: 110-125V AC at 55-65 Hz Output: 12V DC, 1.5A
System Interaction	Supplied by an AC source. Output must be capable of supplying a peak current of 1.5A of current along the shared 12V rail, with an idle draw of approximately 100mA. Full load will be for brief periods of time (when running the stepper motor or a single solenoid valve). Actual peak current is anticipated to be closer to 0.6A, but the supply should be able to cope with unexpected higher draw.
Operator Interaction	Unit will be connected to the wall socket using a standard power supply cable. Surge protectors/breakers will be employed to cut off power in the event of a source-side short. We will use GFCI outlet as protection against liquids around electrical components. DC-side connections will be made via covered screw-terminals.
Hardware/software interface	The unit will be contained in a single aluminum housing. AC will be supplied through a C14 inlet, DC through screw terminals. All DC-side connections will share a common ground for reference. The power supply will maintain a connection with all interfaces shown in the block diagram.

Table 2. Functional Requirements and Specifications of Raspberry Pi 3 Block**.**

Requirements	Specifications
Functional Requirements	Receives user and data inputs of recipes and sends information to Arduino Mega via serial communication.
Performance Requirements	Power requirement of 7-12V DC with a peak current draw of 500mA.
System Interaction	Operates from DC power supply. Must output and accept data input via serial connections
Operator Interaction	Touch screen buttons on face of unit.
Hardware/software interface	Interfaces with user by communicating to LCD, Android Mobile Application, and Arduino Mega.

Table 3. Functional Requirements and Specifications of Arduino Mega Block**.**

Requirements	Specifications
Functional Requirements	Receives data inputs through serial communication with Raspberry Pi 3, and controls outputs and program flow
Performance Requirements	Power requirement of 7-12V DC with a peak current draw of 500mA.
System Interaction	Operates from DC power supply. Must output and accept data input via serial and parallel GPIO connections
Operator Interaction	Direct interaction is not applicable, but receives user input through serial connection.
Hardware/software interface	Interfaces with user by communicating commands to motor /valve controllers and LEDs

Table 4. Functional Requirements and Specifications of LCD Block.

Requirements	Specifications
Functional Requirements	Outputs an array of 5x8 characters, and accepts digital input from control system
Performance Requirements	Power requirement of 3.3V DC input via direct wires from control system GPIO. Amperage functionally negligible.
System Interaction	Operates from DC power supply. Accepts character input over 6 parallel digital data lines
Operator Interaction	Displays characters to convey information to the operator.
Hardware/software interface	6 supply and control pins, as well as 6 data pins.

Table 5. Functional Requirements and Specifications of LED Block.

Requirements	Specifications
Functional Requirements	Illuminate liquor bottles and beverage vessel positions.
Performance Requirements	Lighting patterns show the dispensing modes creating an extra visual element.
System Interaction	Switches at bottle positions and vessel platform are triggered indicating proper placement.
Operator Interaction	Customer places vessel centered on moveable platform, triggering LED.
Hardware/software interface	Raspberry Pi 3 governs light patterns, which give visual indication of drink progress.

Table 6. Functional Requirements and Specifications of Stepper controller and Motor Block.

Requirements	Specifications
Functional Requirements	Stepper system moves conveyer belt back and forth over precise distances as dictated by the system controller.
Performance Requirements	12 VDC, 0.33A input drives the motor's movement, 3.3V signaling used to communicate with system controller.
System Interaction	Interprets movement commands to cycle motor by steps as instructed by the control system. Motor then moves conveyor belt along each step.
Operator Interaction	Moves cup container to next relevant location. No direct interaction with operator.
Hardware/software interface	12V DC power input, 3.3V TTL data input.

Table 7. Functional Requirements and Specifications of Android Mobile Application Block.

Requirements	Specifications
Functional Requirements	Scan PDF417 via API of barcode. Determines if user is of age. Displays menu of choices of beverage depending on age constraint. Home screen with two buttons. Buttons include 'Scan ID' and 'Exit'
Performance Requirements	Access connection to Raspberry Pi's Bluetooth and Wi-Fi capabilities.
System Interaction	Uses Android's barcode scanning API. API uses phone's camera to scan for the barcode. Phone will communicate with Raspberry Pi's servers which holds the database of the beverages.
Operator Interaction	Touchscreen on phone. User will select what to do by touching the 'make beverage' option to the Raspberry Pi. User must have a government issued ID or Driver's License with a barcode on the back side to for the phone to scan.
Hardware/software interface	Android OS system interacts via Android's API and phone will communicate with Raspberry Pi's servers which holds the database of the beverages.

Table 8. Functional Requirements and Specifications of Solenoid Controllers and Actuators Block.

Requirements	Specifications
Functional Requirements	Open and close specified valves to dispense liquid on command.
Performance Requirements	12V, 320mA power supply required to open and close valve. 3.3V signaling used to communicate with system controller.
System Interaction	Begin and cease dispensing specified liquid as dictated by the system controller.
Operator Interaction	No direct interaction, resultant streams of fluid visible.
Hardware/software interface	12V DC power input, 3.3V TTL data input.

7.3 Technical Details of Implementation

7.3.1 Flow Charts

The operational state/program flow through Polytéleia mobile application is presented in Figure 2 This will also be a part of the unit LCD interface with a password option instead of an ID scanner. Additionally, the unit will have a separate menu for setting preferences. The operation state/program flow chart for this can be seen in Figure 3.

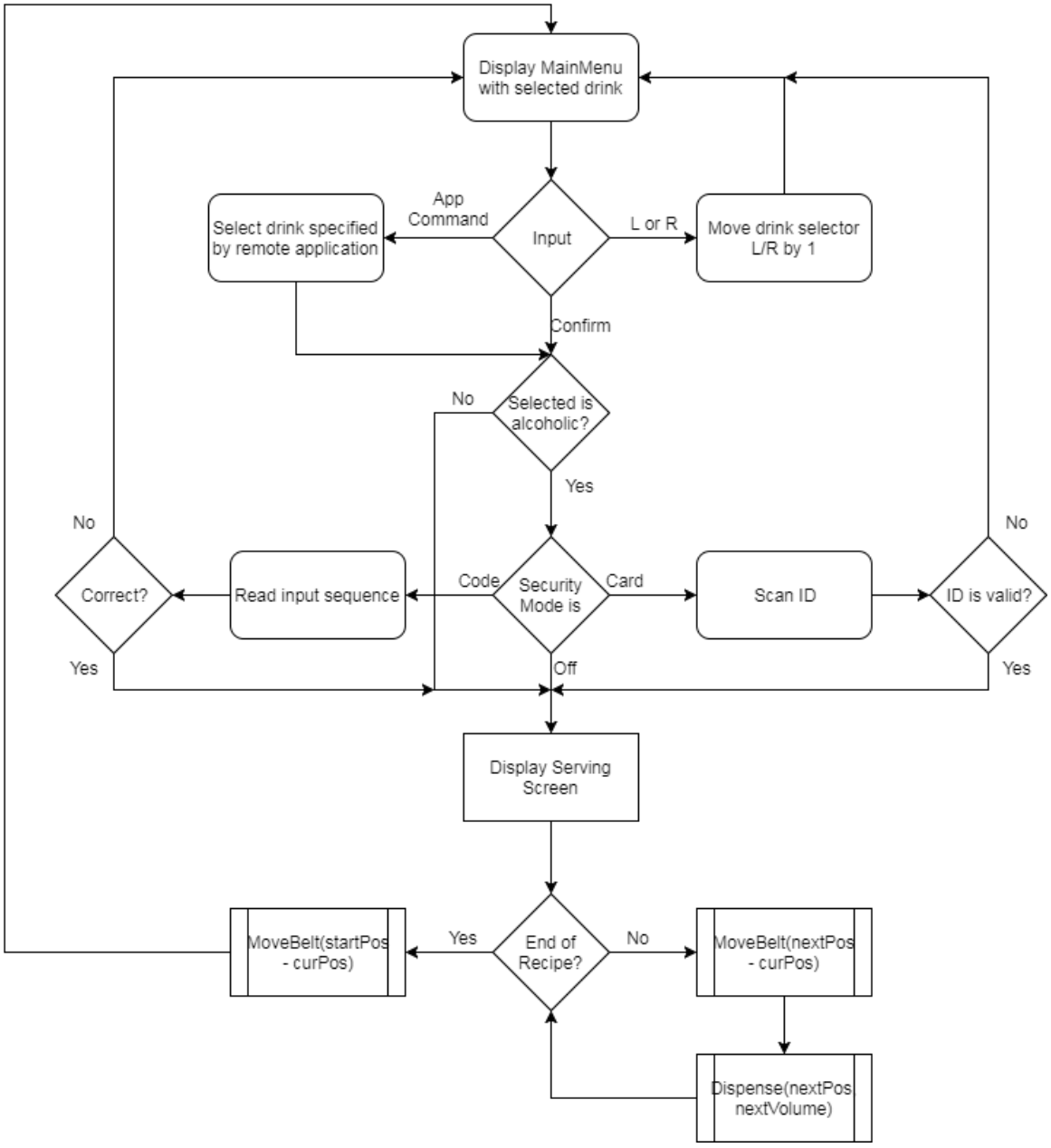

Figure 2. Program Flow Chart for mobile app.

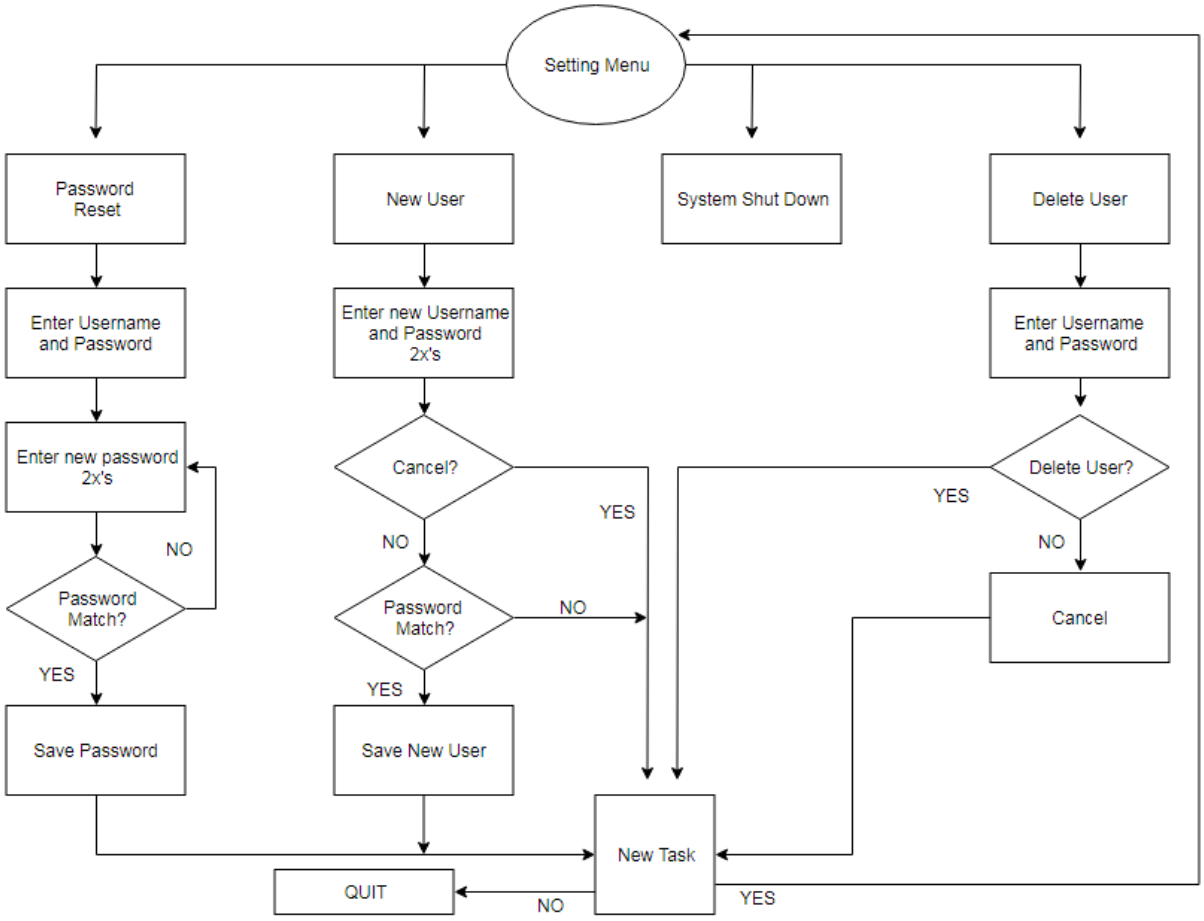

Figure 3. Program Flow Chart for unit settings menu.

7.4 Brief Discussion of other Technical Approaches Considered:

Throughout the design process multiple technical approaches were considered. Originally, the main controller for the system was an Arduino Mega which was going to drive the physical components of the system, as well as, the graphical user interface. When the decision to integrate an Android Mobile Application was made, a Raspberry Pi 3 was added due to its WIFI and Bluetooth capabilities. Additionally, the programming language for the Arduino Mega was not capable of implementing a graphical user interface.

A couple of techniques were discussed for the communication between the two devices. One was having the Arduino read a text file which would have held the recipes for the drinks. The data would have been converted into an array of strings and then executed by the Raspberry Pi. This created memory allocation issues which were too significant to ignore. Serial communication was later implemented as it allowed the Arduino to solely control the unit components, while the Raspberry Pi controlled the graphical user

interface and the Android Mobile Application. The Raspberry Pi simply sends a small array of numbers that contain the dispenser location and the dispensing time for the specific recipe that has been selected.

8. Part List and Budget:

Parts	Unit Quantity	Unit Price	Total
Raspberry Pi 3	1	Owned	$0.00
Gravity Feed Solenoid values	1	$15.99	$63.96
Touchscreen LCD	1	$57.00	$57.00
DC Motor	1	$14.42	$14.42
Wire/Accessories (diodes, phone wire, staples, male-female Wires, electrical wire, heat shrink cable wrap, valve connectors)	1	$96.04	$96.04
Air Pumps (4pack)	1	$16.99	$16.99
Power Supply	1	$17.99	$17.99
Relay board	1	$13.99	$13.99
Peristaltic Pump	1	$6.98	$27.92
Power Converter	1	$9.51	$9.51
Inlet Power Socket	1	$6.99	$6.99
Arduino Mega	1	$38.50	$77.00
Conveyor belt and track	1	Owned	$0.00
Frame material	1	$74.00	$74.00
Plumbing accessories	1	Owned	$0.00
LEDs	1	$6.65	$6.65
Construction supplies (bolts, screws, nuts, stain)	1	$79.64	$79.54
		Total:	$562.00

Table 9. Part and Price Lists.

9. Gantt Chart:

Tasks	Start Date	End Date	Duration
Planning	28-Aug-17	07-Oct-17	40 days
Team formation	28-Aug-17	29-Aug-17	1 days
Project Selection	28-Aug-17	29-Aug-17	1 days
Design ideas	28-Aug-17	06-Oct-17	39 days
Design Phase	28-Aug-17	06-Nov-17	70 days
Research	15-Sep-17	04-Nov-17	50 days
Gather ideas for implementation	15-Sep-17	25-Sep-17	10 days
Research parts	15-Sep-17	04-Nov-17	50 days
Gather parts	08-Dec-17	15-Dec-17	7 days
Aesthetic Design	16-Dec-17	30-Dec-17	14 days
Coding	16-Dec-17	30-Jan-18	45 days
Phone app development	16-Dec-17	15-Jan-18	30 days
Raspberry Pi 3 development	16-Dec-17	30-Jan-18	45 days
Tests	30-Jan-18	04-Feb-18	5 days
Bug fixes	05-Feb-18	19-Feb-18	14 days
Improvements	19-Feb-18	10-Apr-18	50 days
Final testing	05-Mar-18	24-Apr-18	50 days
Final Product	25-Apr-18	26-Apr-18	1 days

Table 10. Design schedule table for project.

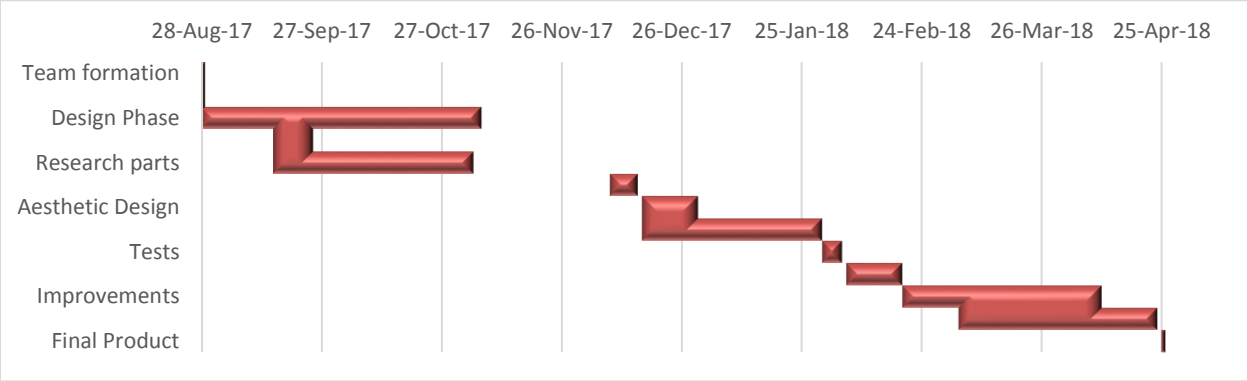

Figure 4. Gantt chart for project.

10. Test Approach:

Component testing was accomplished through command line and physical simulation with Arduino code.

- Track mechanism was tested by tracking the number of ticks the stepper motor drove from front end of the device to the other end. Once the minimum and maximum ticks were recorded the tick location for the dispensers were determined through trial and error.
- The solenoid valves were tested by energizing the valve and measuring the amount of liquid dispensed for an approximated amount of time. From this, the flow rate was calculated and programmed in the code. When attached to the unit the valves were tested again to conclude no further issues would arise due to the angle of the coupling that the valve attaches to. A timing delay was added to account for dripping.
- The air pumps by measuring the amount of liquid dispensed for an approximated amount of time. These values were different than the valves values due to the path diameters for the liquids. From this, the flow rate was calculated and programmed in the code. A timing delay was added to account for dripping.

Testing for the Android Mobile Application was accomplished mainly through command line interaction.

- Age verification was done by scanning the 2D barcode on the back a of state ID. The scanner was tested by simply reading multiple barcodes of users over the age of 21 and users under the age of 21. If the user was of age, the menu would be displayed, otherwise the app would output a message saying, “You are not 21” and would remain on the barcode scanner.
- Ordering drinks were tested by selecting a recipe available on the app menu and making sure that the recipe was executed as it was programmed in the code. Multiple drink recipes of various lengths and dispenser locations were selected to test complete functionality.
- Mobile application connectivity was tested by ordering beverages in several locations away from the unit.

The graphical user interface was tested through Python IDLE and Python shell.

- The GUI was tested by assigning string outputs to the different buttons and selection options. If successful, the string assigned to the specific button being tested would output to the Python IDLE window.
- User authentication was tested by entering several incorrect passwords. Similar passwords and usernames were also used to ensure no access was granted without proper credentials. Additionally, new users cannot be added without a username and password that has already been established.

- Ordering drinks were tested by selecting a recipe available on the adult menu and making sure that the recipe was executed as it was programmed in the code. Multiple drink recipes of various lengths and dispenser locations were selected to test complete functionality.

11.Final Product/Project Results:

The final product that was tested and executed is a success overall. The final design is different than originally intended, but they were all cosmetic issues and nothing to do with functionality. For example, the structural design is meant to have wooden panels to cover the electrical components (i.e. solenoid valves, air pumps), but due to various bottle sizes the compartments are not large enough to enclose them. Additionally, component sizes varied more than anticipated. The structure is unable to elegantly hide the components from plain sight. Due to budgetary constraints, we were limited to four solenoid valves instead of the 9 that were originally specified. To compensate for this issue, we tied the four juice/mixer hoses into our dispensers to be able to demonstrate a larger range of recipes and to prove overall functionality of the device. Besides these minor issues the system works as intended.

Figure 5. Polytéleia structure.

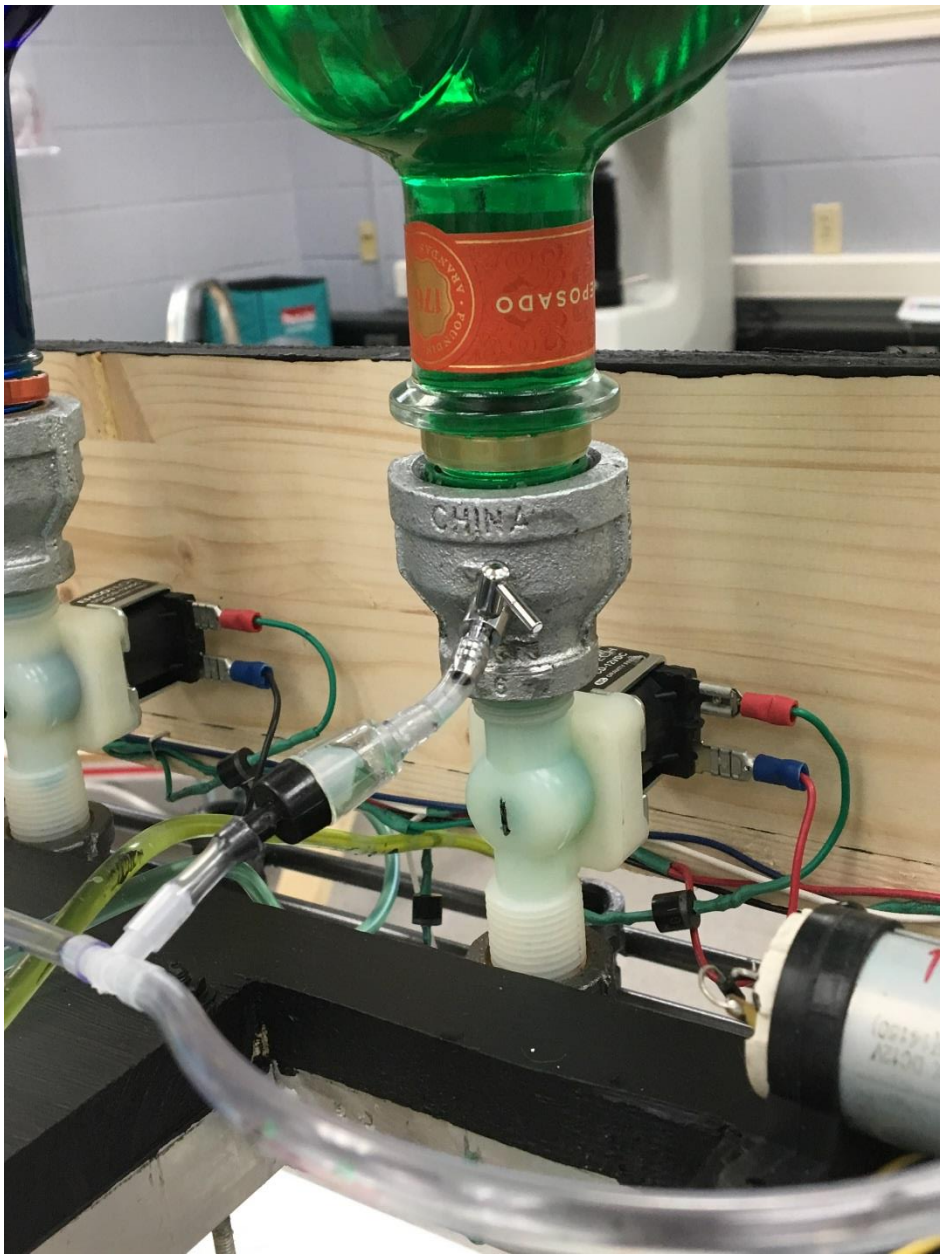

Figure 6. Coupling and gravity feed valve.

The gravity feed solenoid valves were a new component to our final design. The gravity feed valves were necessary because our bottles are upside down. This allowed us to dispense liquid without having to pressurize the bottles. Our original solenoid valves need a minimum psi value to open. The bell couplings attached were added to fit the bottle to the solenoid valve. The couplings were selected specifically, to hold any type of bottle with threads.

Figure 7. Peristaltic air pumps.

The peristaltic air pumps were a new component to the final design. This type of pump was necessary because we needed to successfully pull all the liquid from a vertically positioned container. The pump would then move the liquid through a hose to a designated dispenser location. Originally, we planned to use air pumps to pressurize the containers enough to push the liquid out through a hose. An air hole was drilled in the bottle to avoid compression of the container.

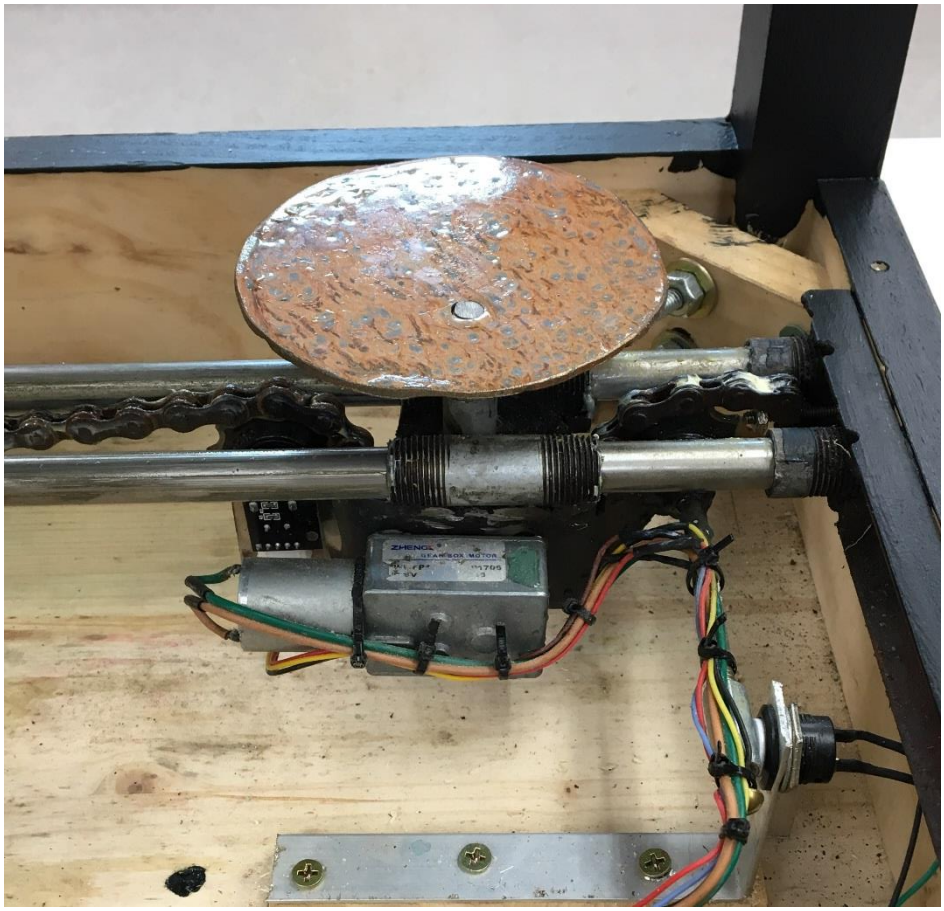

Figure 8. Track mechanism.

The track mechanism consisted of a chain and sprocket drive system. The drive system consisted of 2 steel rods parallel to each end of the device cup. Sleeves known as followers kept the carrier movement confined to 1 degree of freedom. The drive gear is coupled to a 6-volt motor with gear reduction. Through testing we concluded that a 30 RPM geared motor provided a suitable velocity to keep the cup from tipping off the platform during operation. The front most sprocket is coupled to a rotary encoder and the rear most sprocket is coupled to a follower shaft. Special brackets were placed at each end of the track path to prevent the sprockets from binding in the chain if parameters were exceeded.

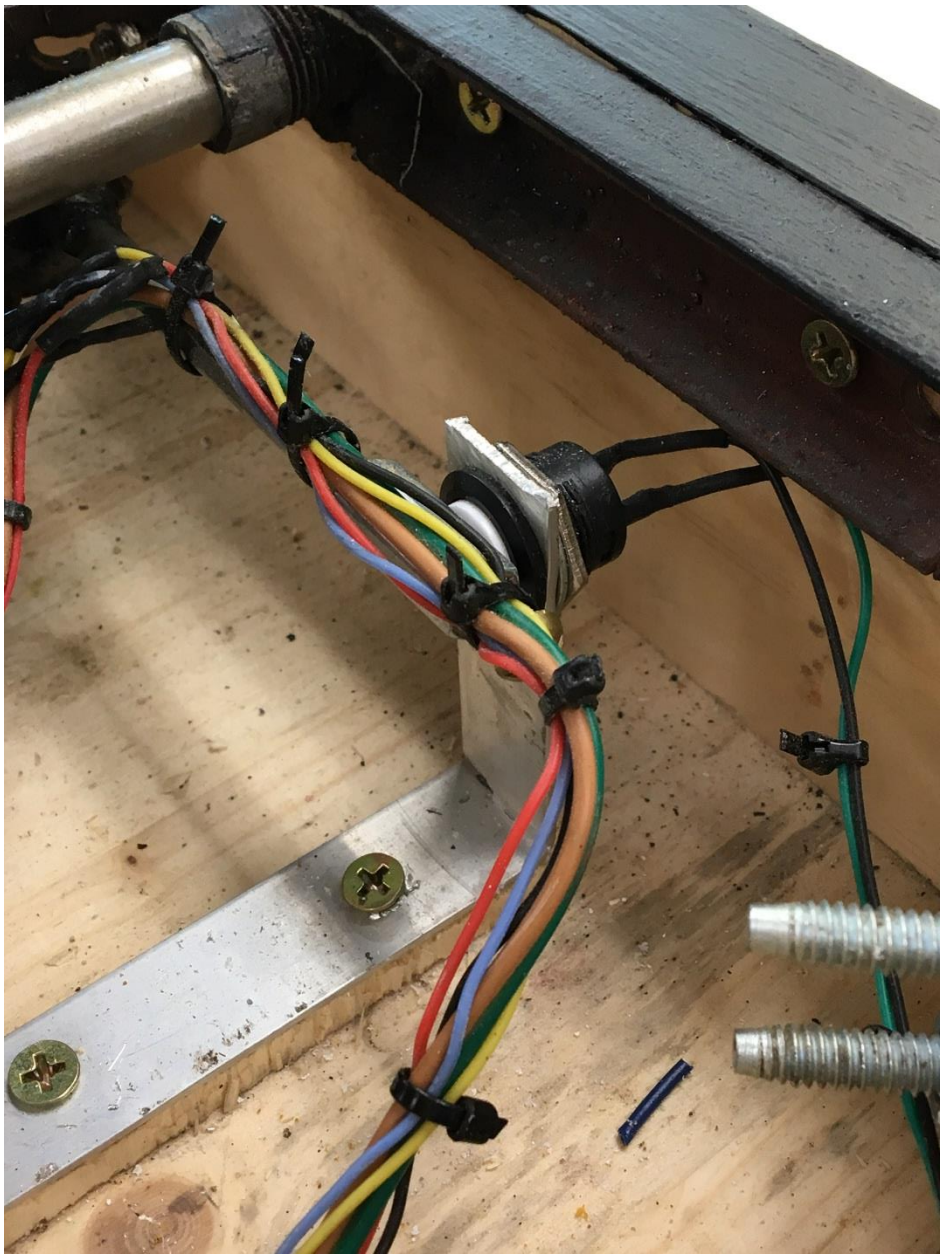

Figure 9. Rotary encoder reset button.

The button shown in Figure 9 was added as a reset for the rotary encoder. As the track mechanism returned to the home position after successfully completing a drink recipe, the button would be pressed and would set the position of the encoder to zero. This was to ensure that the predetermined dispenser locations were maintained accurately. Without this implementation, the encoder would produce skewed results.

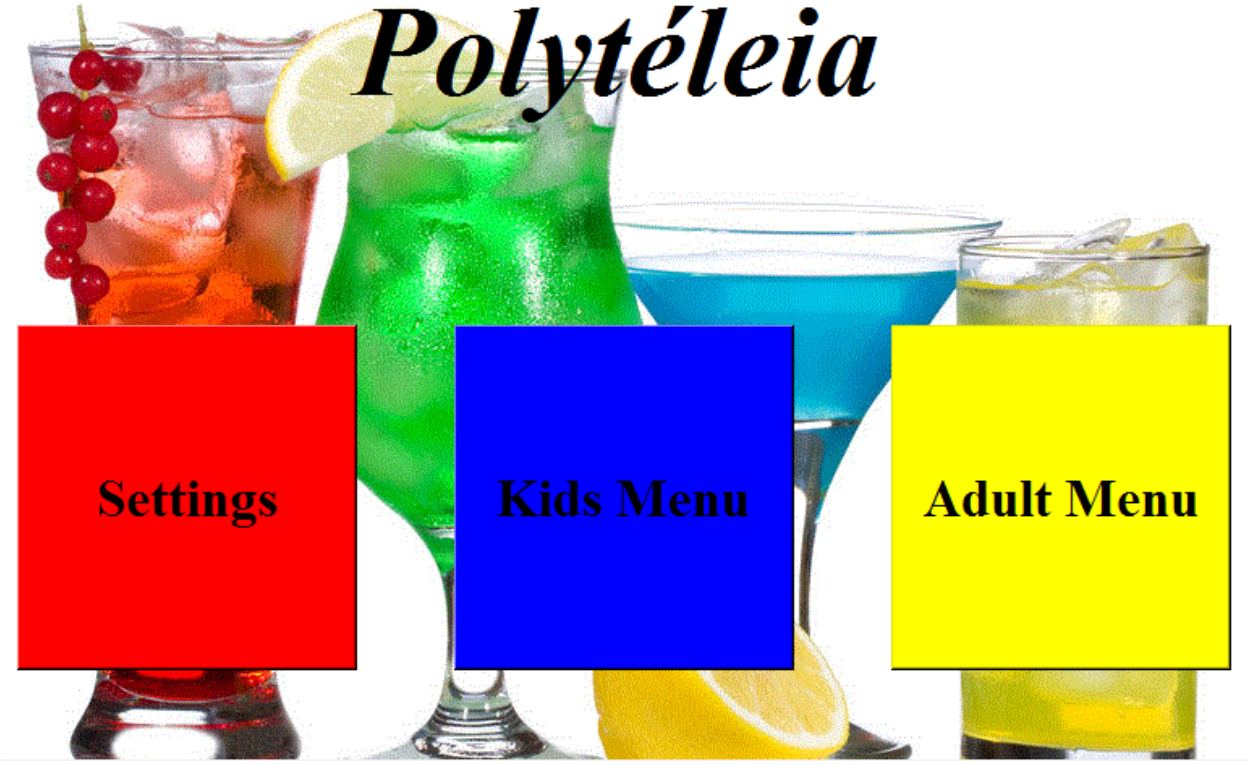

Figure 10. Polytéleia graphical user interface.

The graphical user interface for Polytéleia was programmed in Python 2.7 and tested using the Python Shell and IDLE. Figure 10 shows the home screen for the GUI. The interface was implemented using a seven-inch LCD touchscreen driven by a Raspberry Pi. Each button can be selected via touch or a mouse. They had their own window with a variety of options. Once they were executed the windows would close and would return to the home screen.

Figure 11. GUI adult menu.

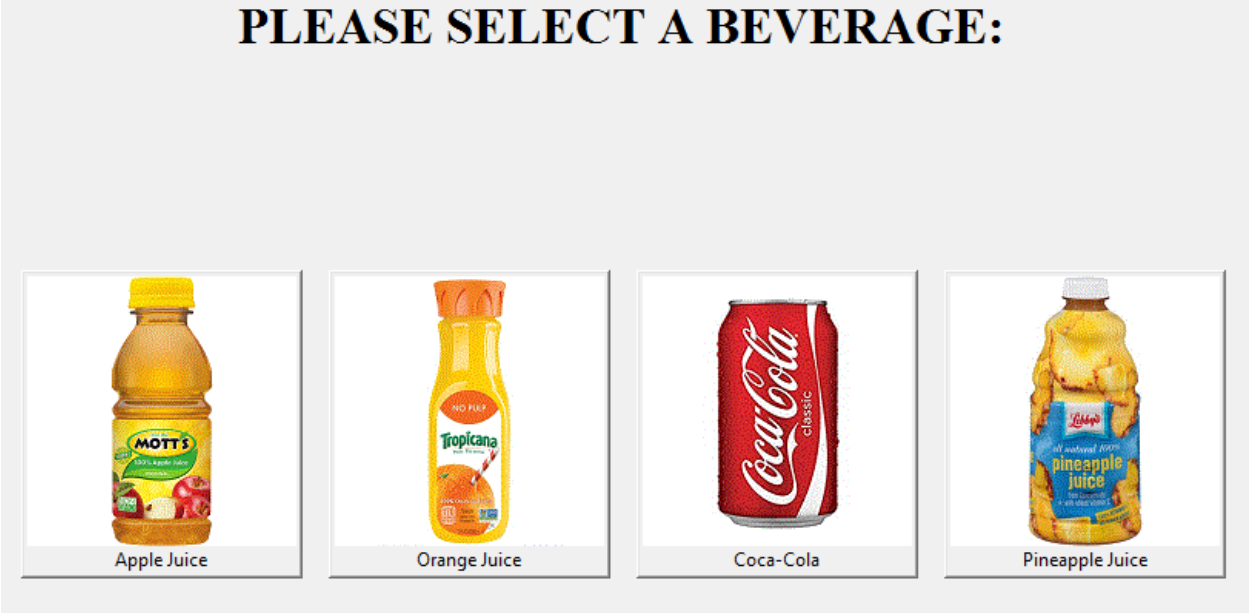

Figure 12. GUI kid's menu.

From the home screen the adult menu selection brings up the window shown in Figure 11. Before this is done the user is required to enter a username and password that has been saved within the system. This was implemented to avoid having users under the age of twenty-one from accessing and ordering from the menu. The adult menu was designed with the intention of holding several drink recipes. A scrollbar was added in the code to allow for this without having to add and open another window. Additionally, the menu has the capabilities of allowing the user to select whether they would like a single shot or a double shot for specific drink recipes. Once a selection is made the adult menu closes and returns to the home screen. Figure 12 shows the kid's menu which has the same functionality as the adult menu. The kid's menu is not password protected since it is limited to beverages that are appropriate for users of any age.

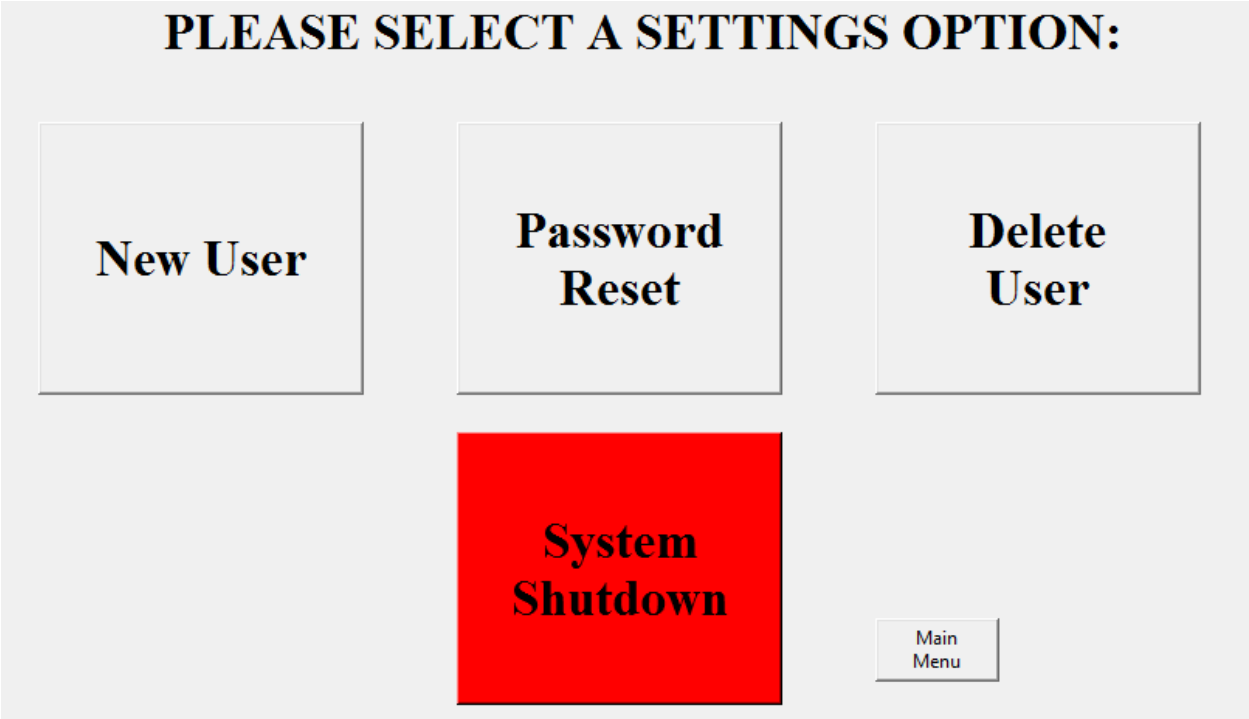

Figure 13. GUI settings menu.

The settings menu options can be seen in Figure 13. The user has the option to create a new user to allow accessibility to the adult menu. When the option is selected, a window opens and asks for a username and password of an established user profile. This was programmed as a safety measure to avoid new users being added and able to access the drink menu. Once the correct credentials have been entered, the menu asks for a new username and password and the profile is saved. The password reset option allows for an established profile to change their original password. When the option is selected, a window opens and asks for the username and password of the profile that requires the password change. The user is prompted to enter the new password in twice, and once the program confirms that they are a match the profile is saved with the updated password. The option to delete a profile allows the user to delete any established profile, when the correct credentials are entered. A second window opens as a warning to show that the

user profile has officially been deleted. The system shutdown option is the only way to completely close the GUI. This was added as a safe way to properly shut down the system.

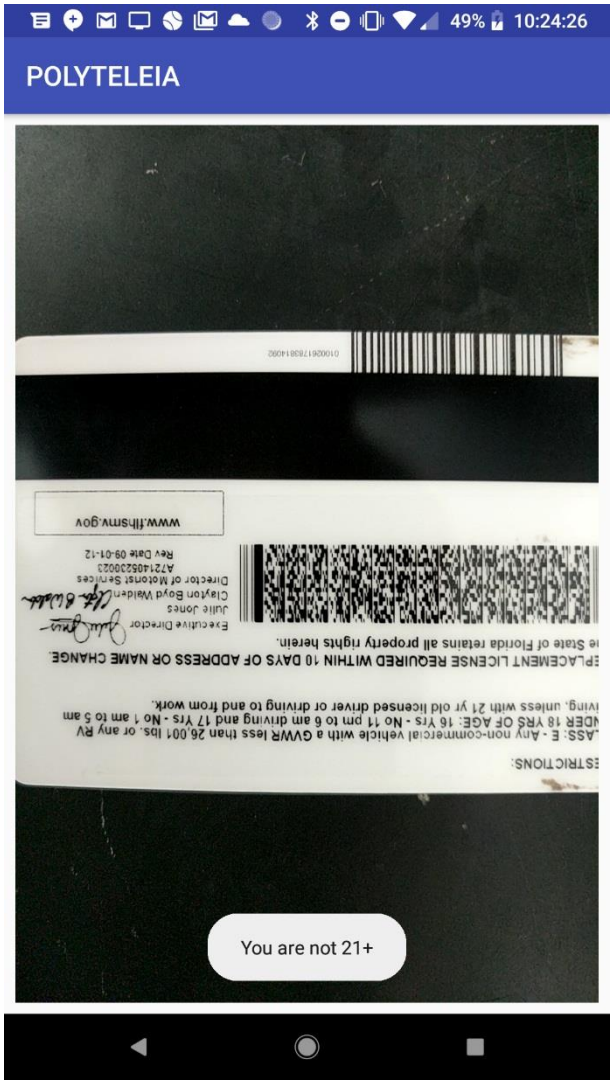

Figure 14. Mobile Application ID rejection.

The Android Mobile Application was designed to allow the user to access the adult drink menu using WIFI or Bluetooth. Since an app can be easily accessible, an ID scanner (barcode scan of the PDF417) was implemented to verify the age of the user. Figure 14 shows the app denying access to the menu and displaying that the user is not of age. When the user is granted access, the application displays a drop-down menu of the drink recipes available. The recipes available match the recipes that are available through the adult menu in the GUI. A signal is then sent to the device and the recipe is processed.

12. Flow Chart of Design Process

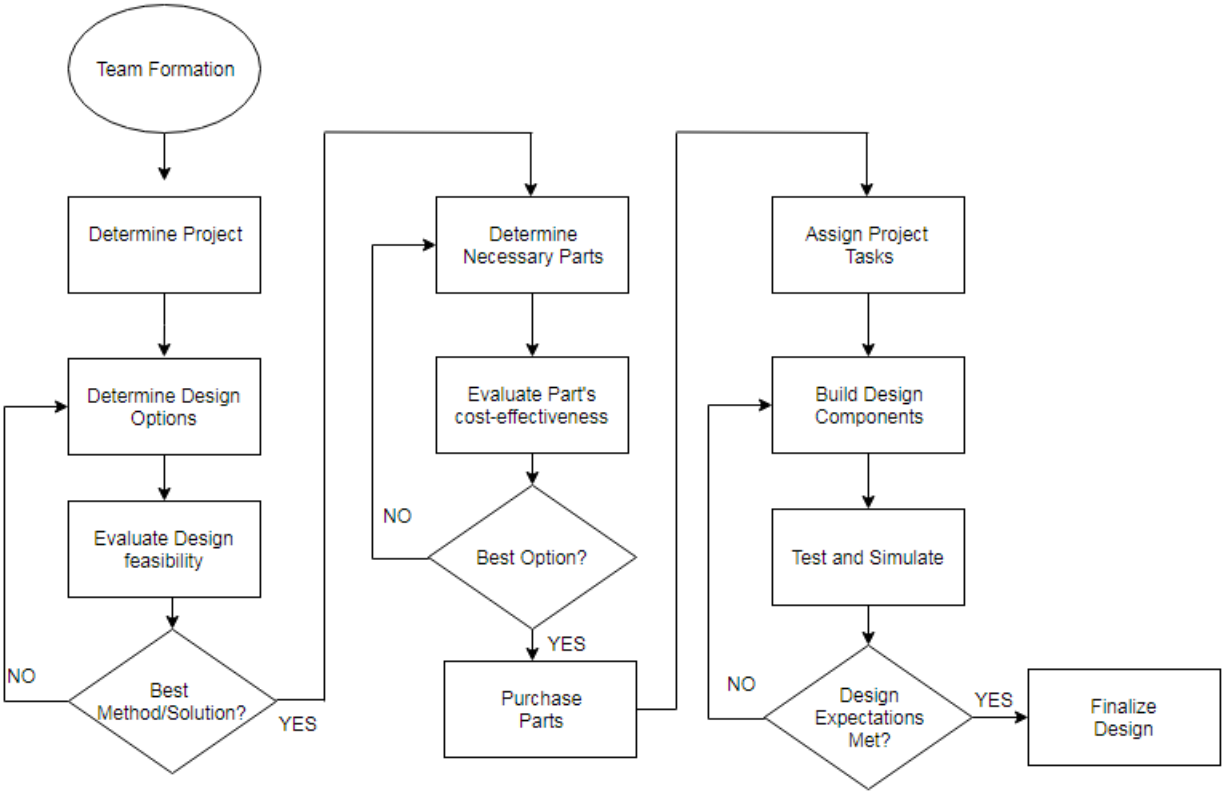

Figure 15. Design Process Flow Chart.

13. Conclusion:

Polytéleia is a standalone automatic bartender. The device receives an order from the user through the LCD touchscreen, or mobile app. The LCD touchscreen grants access to the adult menu once a saved username and password is entered successfully. If the mobile app is used an ID card will be scanned and the user will be granted access if the ID card is of someone 21 and over. Once the order is placed the track mechanism moves the cup horizontally to each specific dispenser location based on the order. Liquid will be dispensed based on recipe and this continues until order is complete. LED's light up to track bottles that are being dispensed and when the drink recipe is being fulfilled and once its completed. The touch screen at the device will have a settings menu that will give the user more access to customizing their device. Whether the device is used in restaurants to eliminate over-pour profit loss, or as a fun luxury entertainment system that can help the host take care of guests, Polytéleia has proven to be a convenient solution for dispensing mixed beverages for commercial and at-home application.

14. Assessment of Math, Science, and Engineering Topics:

Table 11. Assessment of Math, Science and Engineering Topics.

Topics	Annaly Benitez	Taylor Lookabaugh	Matthew Whitlow	Stefan Wheeless
CAD/modelling/simulations	**1**	**1**	**3**	**2**
Chemistry courses	**1**	**1**	**1**	**1**
Circuits	**2**	**2**	**4**	**4**
Communications	**1**	**1**	**1**	**1**
Computer Architecture	**2**	**2**	**2**	**1**
Control Systems	**2**	**2**	**2**	**2**
Digital Design	**2**	**2**	**1**	**1**
Electromagnetic	**1**	**1**	**1**	**1**
Electronics	**2**	**2**	**3**	**4**
Engineering Software tools	**2**	**2**	**2**	**2**
Fluid/Thermodynamic systems	**1**	**1**	**1**	**1**
Image Processing	**1**	**1**	**1**	**1**
Industrial/Manufacturing Engineering	**1**	**1**	**3**	**2**
Material Science	**1**	**1**	**2**	**1**
Math courses	**2**	**2**	**2**	**2**
Mechanical Engineering	**1**	**1**	**4**	**2**
Microprocessors/Interfacing/Embedded Systems	**4**	**4**	**2**	**2**
PC Board	**1**	**1**	**1**	**1**
Project Management	**4**	**1**	**1**	**1**
Physics courses	**1**	**1**	**1**	**1**
Power Engineering	**1**	**1**	**2**	**4**
Programming	**4**	**4**	**4**	**2**
Signal and Systems	**1**	**1**	**1**	**1**
Soldering/welding	**1**	**1**	**2**	**3**
Statics and Dynamics Course	**1**	**1**	**2**	**1**
Statistics	**1**	**1**	**1**	**1**
System integration	**4**	**4**	**4**	**4**
Very Large Scale Integration (VLSI)	**1**	**1**	**1**	**1**

15. Team Activity Report:

Table 12. Team Activity Report – Summary of Contributions by Team Members

	Team Members			
Sections	Annaly Benitez	Taylor Lookabaugh	Matthew Whitlow	Stefan Wheeless
Abstract	X			
1.0 Introduction	X			
2.0 Problem Definition	X			
2.1 Problem/Need	X			
2.2 Intended user(s) and use(s)	X			
2.3 Assumptions and limitations	X			
2.4 Design Objectives	X			
3.0 Design Specifications	X			
4.0 End-Product Description	X			
5.0 Design constraints		X		
5.1 Economical		X		
5.2 Environmental		X		
5.3 Social and Political		X		
5.4 Ethical		X		
5.5 Health and Safety		X		
5.6 Manufacturability and Sustainability		X		
6.0 Engineering Standards			X	
7.0 Technical Approach			X	
7.1 Functional Block diagrams			X	
7.2 Functional Requirements for Blocks			X	
7.3 Technical Details			X	
7.3.1 Flow Charts			X	
7.4 Other Approaches			X	
8 Part List and Budget				X
9 Gantt Chart				X

10. Testing Approach				X
11. Final Product/Project Results				X
12. Flow Chart of Design Process				X
13. Conclusion				X
Bibliography				X

Appendix A:

Annaly Benitez

Home Address:
510 Parkview Unit B
Fort Walton Beach, FL 32547

Email: annaly.benitez@gmail.com
Cell: 509.368.4237

Qualifications

- Computer engineering testing and research, University of West Florida, Fort Walton Beach, FL.
- 2-years of experience in Aircraft Maintenance, 92nd Maintenance Group, Fairchild AFB, WA.
- Bilingual: Fluent in Spanish and English.

Education

- **Bachelors of Science in Computer Engineering** **GPA**: 3.38 **May 2018**
  University of West Florida, Fort Walton Beach, FL

Relevant Experience

Instructional Skills

- MATLAB, C programming, Python, Java, VHDL, and Microprocessor assembly code.
- VLSI design with MOSFET integrated circuits, including proficiency in the DSCH and Microwind2 program.
- Digital logic, MOSFET logic, circuit design and analysis, using Multisim, and LogicWorks.
- Knowledge in proper lab equipment usage, including Tektronix Digital Oscilloscope, Agilent Wave Function Generator, multi-meters.

United States Air Force Crew Chief **Jan 2012 - April 2014**

- Maintained complete accountability of all essential maintenance and aircraft equipment that valued over $5 million.
- Processed multiple equipment and supply orders while communicating with supervision to coordinate timely deliveries.
- Managed and redesigned inventory program saving over $100 thousand in annual budget.
- Completed multiple Air Force wide inspections with outstanding results through teamwork and problem solving skills in a high stress and diverse environment.

University of West Florida Tutor / Facilitator **Jan 2017 - Present**

- Selected by professors to assist electrical and computer engineering students with complicated homework or lab questions that they have.
- Create practice tests, additional homework questions, and conduct exam reviews throughout the semester.
- At instructor's request, hand out assignments, collect assignments, and proctor exams for the students.

Security Clearance

- **Secret**; held while stationed on Fairchild AFB **Jan 2012 - April 2014**

Memberships and Awards

- UWF SGA Emerald Coast – President **Nov 2018 - May 2018**
- IEEE Honors Society, Eta Kappa Nu – President **Mar 2017 - Current**
- UWF Student Government Association – Senior Officer **Apr 2017 - Current**
- UWF Emerald Coast IEEE Organization - Member **May 2016 - Current**
- NDIA Foundation Scholarship **Aug 2017 - Current**
- Stephenson Foundation Scholarship **Aug 2017 - Current**

TAYLOR J LOOKABAUGH

705 Gulf Shores Drive Unit 404
Destin, FL 32541 looktj@gmail.com Cell: 559-341-7354

OBJECTIVE

Seeking a position as Computer Engineer in the technology, manufacturing, or Aerospace industry.

SUMMARY

An EEE certified Computer Engineer with a focus on software and mathematics. Team building proven with applied demonstration during senior project. Areas of expertise include:

Complex mathematics	Software design
Software application	Electrical design
Computer hardware design	Computer hardware application

ACCOMPLISHMENTS

Video Security Design Certificate. Designed secure client spaces with Pelco. Operated CCTVs and honed video security design skills in a laboratory environment.

FFA – SAE Project. Raised and show hogs to sell to market.

Drink Dispenser Machine. Designed a REST API and mobile app to work with a drink dispenser. Primarily alcoholic beverages. Designed mobile app to check a person's ID before making a drink of choice.

EXPERIENCE

Engineering

- Design and build simple to complex circuits.
- Soldering experience with Printed Circuit Boards.
- Embedded systems.
- Programming/software experience.
  - C, C++, VHDL, Assembly, MATLAB, Python, VLSI
- Linux experience July 2006
  - Servers: SSH, Web servers, FTP, MySQL, etc.
  - Administered website for a church.

ACTIVIES

IEEE: Participated in IEEE activities
Tutoring: Helped other students with schoolwork.

EDUCATION

BS, Computer Engineering: *University of West Florida* August 2014 – May 2018
Minor in Computer Science
Pensacola, Florida
AS, Computer Science: *Reedley College* August 2009 – May 2014
Reedley, California
GPA: 3.02

MATTHEW C. WHITLOW

Destin, FL • (731) 412 9324 • mw83@students.uwf.edu

Education

- **B.S Mechanical Engineering | University of West Florida** **2015-2018**
  Minor Electrical engineering – 3.00 GPA
- **A.A. Pre-Engineering | Northwest Florida State College** **2011-2015**
- **Diploma Auto-Diesel Technology | Nashville Auto-Diesel College** **5/2004**
  Additional 3 months course in high performance engines

Professional Experience

Teachers Assistant | University of West Florida **9/2016-Present**

- Selected by professors to grade papers and tutor lower level students
- Assisted professors with lab development and research projects

Leadership | National Society of Leadership and Success **9/2015-2017**

- Selected for induction by professors of University of West Florida
- Attended leadership training programs throughout semester terms
- Obtained graduation regalia upon completion

Auto-Diesel Technician | Service and Dealership industry **2004-2015**

- Served 10 years in industry as a technician
- Solely maintained large fleets
- Developed strong communication skills
- Obtained strong mechanical aptitude
- Advanced in problem solving methods

Qualifications

- Experienced with SolidWorks, Auto-CAD, Multisim, MATLAB, Microsoft Office, and a variety of kinematic simulation software
- Non-Destructive Evaluation
- Modern Product Design
- Engineering and design of unmanned aerial vehicles (UAV)
- Mechatronics and programming in C language
- Proficiency with upper level math and mechanical design

Stefan Wheeless

32 Poquito Rd
Shalimar, FL, 32579

850-420-7125
SBW20@students.UWF.edu
StefanWheeless@gmail.com

Objective

Begin a career in Electrical Engineering – Pursue graduate degree in Systems Engineering

Education

2013-Present Electrical & Computer Engineering Student (Sr.) *University of West Florida* GPA: 3.32

2016/2017 Cybersecurity Scholar – University of West Florida's Center for Cybersecurity
NSA & DHS Center of Excellence (CAE) in Cyber Defense Education

May 2013 Associate of Arts Degree *Northwest Florida State College* GPA:3.49

May 2013 High School Diploma *Collegiate H.S. at NW Florida State College* GPA:3.48 WGPA:4.25

2012-2013 Senior Capstone Project – *Improving School Bus Fuel Efficiency through multidisciplinary approach of technology to assist driver behavior changes to address economic problem.*
Built driver feedback instrument using microcontroller, integrating sensors and datalogger.

Research, Leadership, and Service

Research — Team Lead - UWF Fort Walton Unmanned Systems UAS Project — 2017-Present
- Representative at TeCMEN Industry Day

Scouting — Eagle Scout [Bronze Palm] - Boy Scouts of America — 2000-Present
- now College Reserve adult leader

CHS Robotics Team — Team Programmer and Lead Designer — 2010-2012
- 3rd in US regional of 17 teams – 2012 AFRL & Institute of Navigation Mini Urban Challenge

Saint Simon's Episcopal — Vestry Youth Representative — 2010-2013
- Church Governance of budget, social outreach & education

Volunteer in Father Jack's Soup Kitchen (Providing meals to homeless)

Work Experience

Technician, Developer, IT — ***Emerald Coast Science Center***, Fort Walton Beach, Florida — Summer 2016 - Present
- Maintaining and creating museum exhibits and hardware
- Configuring and installing computer equipment

College Tutor — Independent Tutor for College level Math and Sciences — 2016 - Present

Tech Consultant — Independent consultant for commercial technology applications — 2016 - Present

Tutor — Supervised middle & elementary student summer activities — Summer 2014
- taught chess, robotics, programming, nature activities

Project Planning

Senior Capstone
- Engaged advisory experts in field – Florida Institute of Human and Machine Cognition researcher and Eglin AFB Engineer
- Designed, programmed, calibrated sensors, tested instrument - Analysis of logged data
- Defended research and field work – presented in front of expert panel

Eagle Service Project
- Led fundraising and communications for materials and volunteers
- Designed custom storage unit using - Coordinated two simultaneous construction teams

Name: Annaly Benitez

Major(s):

Computer Engineering

Minor(s):

none

Strengths:

takes initiative, good communicator, organized, good time-management skills

Weaknesses:

expect to much sometimes, may micro-manage and over extend myself,

Electives taken/plan to take:

VLSI, Future energy systems, intro to renewable energy, Plan on taking Mechatronics pattern Recognition

Special skills/abilities/hobbies/background that may be helpful to my team:

Programming in C, experience with microprocessors

Specialty within the team:

programming and designing app processes

Name: Taylor Lookabaugh

Major(s): Computer Engineering

Strengths:

-Presentation skills
-hard worker
-determined
-problem solver
-humility
-resourceful

Weaknesses:

-Reserved/Quiet.
-argumentative at times.
-dexterity

Electives taken/plan to take:

Taken:
-Solid States
-Intro to Renewable Energy
Plan to take:
-Linear Controls
-Pattern Recognition.

Special skills/abilities/hobbies/background that may help team:

-Experience with Linux
-Experience on video surveillance (Pelco).

Specialty within the team:

-Computer Engineering

Name: Matthew Whitlow

Major(s):
Mechanical engineering

Minor(s):
electrical engineering

Strengths:
outgoing, creative, people person
Strong Mechanical skills, motivated

Weaknesses:
Red head temper, Stubborn, hyper
difficult to understnd at times.

Electives taken/plan to take:
Manufacturing processes, Non-destructive eval,

Special skills/abilities/hobbies/background that may be helpful to my team:
Mechanical background, Auto-diesel technology

Specialty within the team:
Mechanical design

Name: Stefan Wheeless

Majors: Electrical Engineering, Computer Engineering

Minors: None

Strengths: Leadership experience (Eagle scouts, Unmanned systems), Effective System design, Can work well in changing conditions, effective problem-solving skills

Weaknesses: Time management facing large quantity of different responsibilities, diligence facing monotonous tasks without some outside form of engagement

Electives: Unmanned systems engineering

Special skills: Drafting/CAD, Simple modeling, circuit design, programming, some woodworking skills, 3D Printing

Specialty within team: Undetermined.

Appendix B:

Main Component Code

```
//Release Version 4/18/18

//program fully functional
//for liquid juice dispensers call to D10, D11, D12, D13 but are physically
located at D3, D5, D7, D9. That's where the cup will go to fill when called
D10 through D13

//edit out Serial.Begin(9600) to run without debug console being open
//command input format (number of ingredients, station of ingredient 1,
ounces of ingredient 1, station of ingredient 2,....)
//example: 2, 3, 1, 5, 1 this goes to station 3 pours 1 ounce then goes to
station 5 pours 1 ounce

//spare matrix location 14 and 15. to turn on Spare15 write PORTA=123; for
Spare14 write PORTA=125;

void LED(int color);                          //control led strip lights
green, blue and red
void motorGoToDispenser(int station);        //insert dispenser number into
function, returns when position is reached
void cupPosition(void);                       //function reads and updates
encoder value
void motorGoToPos(int location);              //function for motor driver
void motorReturn(void);                       //returns to start position and
sets position to zero
void motorDropOff(void);                      //carries cup to end for pick up
void dispenser(int station, float Oz);     //function takes dispenser number
and desired valve on time in milliseconds, returns when dispensed

    //rotary encoder
 int pinA = 3;  // Connected to CLK on KY-040   blue wire
 int pinB = 4;  // Connected to DT on KY-040    yellow wire
 int encoderPosCount = 0;
 int pinALast;
 int aVal;
 boolean bCW;

      //dispensors
 //arduino to relay pinout
//port# pin#--relay#
//PA0    22     I8
//PA1    23     I7
//PA2    24     I6
//PA3    25     I5
//PA4    26     I4
//PA5    27     I3
//PA6    28     I2
//PA7    29     I1

void setup() {
    Serial.begin(9600);

  //dispensors
  DDRA=0xFF; //configure port A as output. logic 0 energizes relay
```

```
  PORTA=255; //pull up to keep relays off

  //motor output                                    green motor wire goes to
center pole relay 1
  DDRB=255; //configure port B as output          brown wire center pole of
relay 2
                                                 //white strip power wire to
norm closed relay 1, jump to norm closed relay 2
  //rotary encoder
   pinMode (pinA,INPUT);                          //relay control, pin 52 to
I1 (relay 1) and pin 53 to I2 (relay 2)
   pinMode (pinB,INPUT);
   pinALast = digitalRead(pinA);


  //home switch to reset encoder                          momentary switch,
green wire to pin 37 and black to ground
  DDRC=0;     //configure port C as input
  PORTC=0xFF; //turn on pullup resistors

  motorReturn(); //bring motor to home position

  DDRL=0xFF; //configure Port L as output
  PORTL=0; //pull up to keep relays off
  LED(1);   //turn on idle blue led
  int L=0;  //timer to switch led back to idle
}
int L=0;  //timer to switch led back to idle
void loop() {



   while (Serial.available() > 0) {

    L=0; //reset led idle timer when drink request is made

     float lineup[15]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

        int N = Serial.parseInt();

           Serial.print(N);Serial.println(" ingredients");

        N=N*2;

            for(int i=0;i<N;i=i+2){
              lineup[i]=Serial.parseInt();
              lineup[i+1]=Serial.parseInt();
             }

            for(int i=0;i<N;i=i+2){
              motorGoToDispenser(lineup[i]);
              dispenser(lineup[i],lineup[i+1]);
              Serial.print("motor goes to station
");Serial.println(lineup[i]);
              Serial.print("dispensing
");Serial.print(lineup[i+1]);Serial.println(" ounces");
```

```
            }
      Serial.println(" ");
      N=0;
      motorReturn();      //drink complete take cup back to user

  }

  delay(200);
  Serial.println(L);
  L++;
  if(L==10){   // user has 8 seconds to remove cup, system goes back to idle
state blue led
    LED(1);  //switch led back to idle
    Serial.println("idele ");
    L=0;
  }

}
void motorGoToDispenser(int station){

  switch(station){

    case 1:
    motorGoToPos(6);    //numbers inside function are the encoder ticks
measured from start postion to directly under dispenser listed in case
    break;                //for this case, the motor comes on, when the
encoder reads 6 ticks the motor turns off

    case 2:
    motorGoToPos(25);
    break;

    case 3:
    motorGoToPos(50);
    break;

    case 4:
    motorGoToPos(76);
    break;

    case 5:
    motorGoToPos(100);
    break;

    case 6:
    motorGoToPos(123);
    break;

    case 7:
    motorGoToPos(150);
    break;

    case 8:
    motorGoToPos(170);
    break;

    case 9:
```

```
    motorGoToPos(195);
    break;

    case 10:
    motorGoToPos(50);
    break;

    case 11:
    motorGoToPos(100);
    break;

    case 12:
    motorGoToPos(150);
    break;

    case 13:
    motorGoToPos(195);
    break;

  }

} //end motorGoToDispenser

void motorGoToPos(int location){      //this function determines if motor
needs to go forward or backward to reach destination

LED(2); // cup moving turn on green led

  if(encoderPosCount<location){

      while(encoderPosCount<location){      //motor forward
        cupPosition();
        PORTB=1;
        }

  }

  else if(encoderPosCount>location){      //motor backward

      while(encoderPosCount>location){
        cupPosition();
        PORTB =2;
      }
  }

PORTB=3;        //motor stop

} //end motorGoToPos

void motorReturn(void){     // function returns motor to home postion and
resets encoder

LED(4); //drink complete turn on blue and red led

  while(PINC==255){
    cupPosition();
    PORTB=2;
```

```
    //Serial.println(PINC);
    delay(10);
  }
   PORTB=3;
   Serial.println("off");
   encoderPosCount=0;
   delay(100);

} //end motorReturn

void motorDropOff(void){      //this function carries the cup to the end of
the deck which is 203 encoder clicks from start to end

  while(encoderPosCount<201){
    cupPosition();
    PORTB=1;
  }

  PORTB=3;
  delay(5000);

} //end motorDropOff

void cupPosition(void){                          //keeps up with rotary
encoder postion

   aVal = digitalRead(pinA);

   if (aVal != pinALast){

       if (digitalRead(pinB) != aVal) {  // Means pin A Changed first - We're
Rotating Clockwise
          encoderPosCount ++;
          bCW = true;
         }
       else {// Otherwise B changed first and we're moving CCW
          bCW = false;
          encoderPosCount--;
         }
     Serial.print("Postion  ");
     Serial.println(encoderPosCount);

   }

   pinALast = aVal;

} //end cupPosition

//juice pump flow rate is 0.3422 ounces per second
 void dispenser(int station, float Oz){      //main function            this
function turns on the dispenser of choice for timeon amount of time

      LED(3); //dispensing turn on green and red led

      float timeOn=0;
      float PumpOn=0;
```

```
     timeOn=Oz*1000;         //flow rate constant to determin time on with
respect to ounces needed
     PumpOn=Oz/0.3442;       //flow rate for juice pumps
     PumpOn=PumpOn*1000;


  switch(station){               //relay #1 is driven by I8. relays 1,2,3
provide power to matrix. remainder relays provide ground.
                                 //when I8 is low 0v relay 1 gets energized
    case 1:
    PORTA=246;            //solenoids on PortA
    PORTB=4;              //airlock airpump on PortB
    delay(timeOn);
    break;

    case 2:
    PORTA=238;
    PORTB=4;
    delay(timeOn);
    break;

    case 3:
    PORTA=222;
    PORTB=4;
    delay(timeOn);
    break;

    case 4:
    PORTA=245;
    PORTB=4;
    delay(timeOn);
    break;

    case 5:
    PORTA=237;
    PORTB=4;
    delay(timeOn);
    break;

    case 6:
    PORTA=221;
    PORTB=4;
    delay(timeOn);
    break;

    case 7:
    PORTA=243;
    PORTB=4;
    delay(timeOn);
    break;

    case 8:
    PORTA=235;
    PORTB=4;
    delay(timeOn);
    break;
```

```
    case 9:
    PORTA=219;
    PORTB=4;
    delay(timeOn);
    break;

    case 10:  //dispensor relocated to D3
    PORTA=187;
    delay(PumpOn);
    break;

    case 11:   //dispensor relocated to D5
    PORTA=126;
    delay(PumpOn);
    break;

    case 12:  //dispensor relocated to D7
    PORTA=189;
    delay(PumpOn);
    break;

    case 13:   //dispensor relocated to D9
    PORTA=190;
    delay(PumpOn);
    break;
  }
  PORTB=0;
  PORTA=255;

  delay(1000); //drip time
} //end dispenser

void LED(int color){

    switch(color){

        case 1: // blue for idle
        PORTL=0;//255;
        break;

        case 2: //green for moving
        PORTL=1;//254;
        break;

        case 3: //green and red for dispensing
        PORTL=3;//252;
        break;

        case 4: //done blue red plus delay time
        PORTL=2;//253;
        break;
  }
}
```

Android Mobile Application Code

```
package capstone.myapplication;

import android.content.Intent;


public class DrinksActivity extends AppCompatActivity {
    Spinner spinner;
//    Intent intent = getIntent();
//    String message = intent.getStringExtra(EnterURIActivity.EXTRA_MESSAGE);
    String URL_test;
    ArrayList<String> DrinkName;
    ArrayList<String> DrinkID;
    private Button viewBtn;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_drinks);
        final Button viewBtn = (Button) findViewById(R.id.button3);

        Intent intent = getIntent();
        String message =
intent.getStringExtra(EnterURIActivity.EXTRA_MESSAGE); //gets user's input
from EnterURIActivity
        final String URL="http://" + message + "/drinks"; //combines to
complete the API's URI
        URL_test = URL;

        DrinkName=new ArrayList<>();
        DrinkID=new ArrayList<>();
        spinner=(Spinner)findViewById(R.id.spinner);
        loadSpinnerData(URL);
        spinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> adapterView, View view,
int i, long l) {
                String drinkname=
spinner.getItemAtPosition(spinner.getSelectedItemPosition()).toString();

Toast.makeText(getApplicationContext(),drinkname,Toast.LENGTH_LONG).show();
                int id_num = spinner.getSelectedItemPosition() + 1;
                final String p_url = URL + "/" + id_num; //makes a post url
based on spinner's selection
                viewBtn.setOnClickListener(new OnItemClickListener(p_url));
//sets the button
                Log.d("json_result", "confirm json: " + p_url); //debug msg
        }

            @Override
            public void onNothingSelected(AdapterView<?> adapterView) {
                // DO Nothing here
            }
        });
```

```
    }

    @Override
    public void onPause(){
        System.exit(0);
        super.onPause();
    }

    private class OnItemClickListener implements View.OnClickListener {//sets
up button's click to send POST request based on arg.
        RequestQueue
requestQueue=Volley.newRequestQueue(getApplicationContext());
        private String post_url;

        OnItemClickListener (String position) {
            post_url = position;
        }

        @Override
        public void onClick(View arg0) {

            final String post_url_real = "http://" + URL_test + "/drinks/" +
post_url;
            Log.d("Title at row" + post_url, "id");

            Map<String, String> jsonParams = new HashMap<String, String>();
            jsonParams.put("param1", "1");
            JsonObjectRequest postRequest = new JsonObjectRequest(
Request.Method.POST, post_url,

                    new JSONObject(),
                    new Response.Listener<JSONObject>() {
                        @Override
                        public void onResponse(JSONObject response) {
                            Toast.makeText(getApplicationContext(),"Making
drink",Toast.LENGTH_LONG).show();
                        }
                    },
                    new Response.ErrorListener() {
                        @Override
                        public void onErrorResponse(VolleyError error) {
                            //   Handle Error
                            Log.e("LOG", error.toString());

Toast.makeText(getApplicationContext(),"Error!",Toast.LENGTH_LONG).show();
                        }
                    }) {
                @Override
                public Map<String, String> getHeaders() throws
AuthFailureError {
                    HashMap<String, String> headers = new HashMap<String,
String>();
                    headers.put("Content-Type", "application/json;
charset=utf-8");
                    headers.put("User-agent",
System.getProperty("http.agent"));
                    return headers;
```

```
                }
            };
            requestQueue.add(postRequest);
        }}

    private void loadSpinnerData(String url) {
        RequestQueue
requestQueue=Volley.newRequestQueue(getApplicationContext());
        StringRequest stringRequest=new StringRequest(Request.Method.GET,
url, new Response.Listener<String>() {
            @Override
            public void onResponse(String response) {
                try{
                    JSONObject jsonObject=new JSONObject(response);
                    if(jsonObject.getInt("success")==1){
                        JSONArray
jsonArray=jsonObject.getJSONArray("drinks");
                        for(int i=0;i<jsonArray.length();i++){
                            Log.d("json_result", "confirm json: " +
jsonArray.length());
                            JSONObject
jsonObject1=jsonArray.getJSONObject(i);
                            Log.d("json_result", "confirm json: " +
jsonObject1.getString("drink_name"));
                            String
drinkname=jsonObject1.getString("drink_name");
                            String drinkid=jsonObject1.getString("drink_id");
                            DrinkName.add(drinkname);
                            DrinkID.add(drinkid);
                        }
                    }
                    spinner.setAdapter(new
ArrayAdapter<String>(DrinksActivity.this,
android.R.layout.simple_spinner_dropdown_item, DrinkName));
                }catch (JSONException e){e.printStackTrace();}
            }
        }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                error.printStackTrace();
            }
        });

        int socketTimeout = 30000;
        RetryPolicy policy = new DefaultRetryPolicy(socketTimeout,
DefaultRetryPolicy.DEFAULT_MAX_RETRIES,
DefaultRetryPolicy.DEFAULT_BACKOFF_MULT);
        stringRequest.setRetryPolicy(policy);
        requestQueue.add(stringRequest);
    }


}
```

```
   package capstone.myapplication;

   import android.content.Intent;
```

```
   public class EnterURIActivity extends AppCompatActivity {
       public       static       final       String       EXTRA_MESSAGE       =
"capstone.myapplication.MESSAGE";
       @Override
       protected void onCreate(Bundle savedInstanceState) {
           super.onCreate(savedInstanceState);
           setContentView(R.layout.activity_enter_uri);
       }


       public void sendMessage(View view) {
           Intent intent = new Intent(this, DrinksActivity.class);
           EditText editText = (EditText) findViewById(R.id.editText);
           String message = editText.getText().toString();
           intent.putExtra(EXTRA_MESSAGE, message);
           startActivity(intent);
       }
}

   package capstone.myapplication;

   import android.content.Intent;
   import android.support.v7.app.AppCompatActivity;
   import android.os.Bundle;
   import android.view.View;

   public class MainActivity extends AppCompatActivity {

       @Override
       protected void onCreate(Bundle savedInstanceState) {
           super.onCreate(savedInstanceState);
           setContentView(R.layout.activity_main);
       }

       public void ScanIDCard(View view) {//sets the button to start a new
activity on click.
           Intent intent = new Intent(this, ScanIDActivity.class);
           startActivity(intent);
       }
}

   package capstone.myapplication;

   import android.Manifest;
   import android.app.Activity;

   public class ScanIDActivity extends AppCompatActivity {
       private static final int MY_PERMISSIONS_REQUEST_CAMERA = 2;
       Activity thisActivity = this;
       SurfaceView cameraPreview;
       TextView barcodeResult;

       @Override
       protected void onCreate(Bundle savedInstanceState) {
           super.onCreate(savedInstanceState);
           setContentView(R.layout.activity_scan_id);
```

```
            cameraPreview = (SurfaceView) findViewById(R.id.camera_view);
            int       rc       =       ActivityCompat.checkSelfPermission(this,
Manifest.permission.CAMERA);
            if(rc == PackageManager.PERMISSION_GRANTED) {
                CreateCameraSource();
            } else {
                if  (!ActivityCompat.shouldShowRequestPermissionRationale(this,
Manifest.permission.CAMERA)) {
                    showMessageOKCancel("You need to allow access to Camera",
                            new DialogInterface.OnClickListener() {
                                @Override
                                public void onClick(DialogInterface dialog, int
which) {

ActivityCompat.requestPermissions(ScanIDActivity.this,                    new
String[]{Manifest.permission.CAMERA}, MY_PERMISSIONS_REQUEST_CAMERA);
                                }
                            });
                    return;
                }
            }
        }

        private         void         showMessageOKCancel(String         message,
DialogInterface.OnClickListener okListener) {
            new AlertDialog.Builder(ScanIDActivity.this)
                    .setMessage(message)
                    .setPositiveButton("OK", okListener)
                    .setNegativeButton("Cancel", null)
                    .create()
                    .show();
        }

        private void CreateCameraSource() {
            BarcodeDetector           barcodeDetector           =           new
BarcodeDetector.Builder(this).setBarcodeFormats(Barcode.PDF417).build();
            final  CameraSource  cameraSource  =  new  CameraSource.Builder(this,
barcodeDetector)
                    .setAutoFocusEnabled(true)
                    .setRequestedPreviewSize(1600, 1024)
                    .build();

            cameraPreview.getHolder().addCallback(new  SurfaceHolder.Callback()
{
                @Override
                public void surfaceCreated(SurfaceHolder surfaceHolder) {
                    try {
                        if
(ActivityCompat.checkSelfPermission(ScanIDActivity.this,
Manifest.permission.CAMERA) != PackageManager.PERMISSION_GRANTED) {
                            // TODO: Consider calling
                            //    ActivityCompat#requestPermissions
                            // here to request the missing permissions, and then
overriding
                            //     public  void  onRequestPermissionsResult(int
requestCode, String[] permissions,
```

```
                        //                                          int[]
grantResults)
                        // to handle the case where the user grants the
permission. See the documentation
                        // for ActivityCompat#requestPermissions for more
details.
                        return;
                    }
                    cameraSource.start(cameraPreview.getHolder());
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }

            @Override
            public void surfaceChanged(SurfaceHolder surfaceHolder, int i,
int i1, int i2) {

            }

            @Override
            public void surfaceDestroyed(SurfaceHolder surfaceHolder) {
                cameraSource.stop();
            }
        });

        barcodeDetector.setProcessor(new Detector.Processor<Barcode>() {
            @Override
            public void release() {

            }

            @Override
            public   void   receiveDetections(Detector.Detections<Barcode>
detections) {
                final        SparseArray<Barcode>        barcodes        =
detections.getDetectedItems();
                for (int i=0; i<barcodes.size();i++) {
                    int key = barcodes.keyAt(i);
                    final Barcode barcode = barcodes.get(key);
   //                   barcodeResult.setText(barcode.format);
   //                   invalidate();
                    Log.d("barcode_result",    "Barcode    Format:    "    +
barcode.format);
                    Log.d("barcode_result",     "Barcode     RAW:     "     +
barcode.driverLicense.birthDate);
                    SimpleDateFormat          sdf          =          new
SimpleDateFormat("yyyyMMdd");
                    Date birthdate_check = null;
                    try {
                        birthdate_check                                    =
sdf.parse(barcode.driverLicense.birthDate);
                    } catch (ParseException e) {
                        e.printStackTrace();
                    }
                    Calendar cal = Calendar.getInstance();
                    cal.add(Calendar.YEAR, -21);
```

```
                        Date age_check = cal.getTime();
                        sdf.format(age_check);
                        Log.d("barcode_result",    "confirm    age:    "    +
birthdate_check);
                        if(!birthdate_check.after(age_check)) {//confirms >=21
and starts EnterURIActivity class.
                            Intent  intent  =  new  Intent(ScanIDActivity.this,
EnterURIActivity.class);
                            startActivity(intent);
                        }
                        else
                        {//presents message if age is <21
                            ScanIDActivity.this.runOnUiThread(new Runnable() {
                                @Override
                                public void run() {

Toast.makeText(getApplicationContext(),"You              are              not
21+",Toast.LENGTH_LONG).show();
                                }
                            });
                        }


                        break;

                    }

                }


            });
        }
}
```

API Implementation Code

```
#!/usr/bin/env python
from flask import Flask
from flask import g
from flask import Response
from flask import request
from flask import jsonify
import json
import MySQLdb
from pprint import pprint
import serial
import time

app = Flask(__name__)

@app.before_request
def db_connect():
  g.conn = MySQLdb.connect(host='localhost',
                           user='looktj',
                           passwd='password',
                           db='drinks')
  g.cursor = g.conn.cursor()

@app.after_request
def db_disconnect(response):
  g.cursor.close()
  g.conn.close()
  return response

def query_db(query, args=(), one=False):
  g.cursor.execute(query, args)
  rv = [dict((g.cursor.description[idx][0], value)
  for idx, value in enumerate(row)) for row in g.cursor.fetchall()]
  return (rv[0] if rv else None) if one else rv

@app.route("/")
def hello():
  return "Hello World!"

@app.route("/drinks", methods=['GET'])
def drinks():
  result = query_db("SELECT drink_id,drink_name,drink_path FROM drinks")
  data = json.dumps(result)
  resp = Response(data, status=200, mimetype='application/json')
  return jsonify({"success": 1, "drinks": result})

@app.route("/add", methods=['POST'])
def add():
  req_json = request.get_json()
  pprint(req_json['drink_name'])
  g.cursor.execute("INSERT INTO drinks (drink_name, drink_path) VALUES (%s,
%s)", [req_json['drink_name'], req_json['drink_path']])
  g.conn.commit()
  pprint(req_json)
  resp = Response("Updated", status=201, mimetype='application/json')
```

```
     return  jsonify({  "drink_name":  req_json['drink_name'],  "drink_path":
req_json['drink_path'] })

   @app.route("/drinks/<int:drink_id>", methods=['PUT'])
   def update_drink(drink_id):
       req_json = request.get_json()
       pprint(req_json)
       g.cursor.execute("UPDATE drinks SET drink_name = %s, drink_path = %s
WHERE  drink_id  =  %s",  [req_json['drink_name'],  req_json['drink_path'],
[drink_id]])
       g.conn.commit()
       return     jsonify({     "drink_id":     drink_id,     "drink_name":
req_json['drink_name'], "drink_path": req_json['drink_path'] })

   @app.route("/drinks/<int:drink_id>", methods=['POST'])
   def create_drink(drink_id):
       g.cursor.execute("SELECT drink_path from drinks WHERE drink_id = %s",
[drink_id])
       row = g.cursor.fetchone()
       print row[0]
       ser = serial.Serial('/dev/ttyACM0', 9600)
       time.sleep(1)
       time.sleep(1)
       print(ser.name)
       ser.write(row[0] + "\n")
       ser.close
       resp = Response("Tested", status=201, mimetype='application/json')
       return jsonify({"result": True})


   @app.route("/drinks/<int:drink_id>", methods=['DELETE'])
   def delete_drink(drink_id):
       g.cursor.execute("DELETE FROM drinks WHERE drink_id=%s", [drink_id])
       g.cursor.execute("SET @num := 0")
       g.cursor.execute("UPDATE drinks SET drink_id = @num := (@num+1)")
       g.cursor.execute("alter table drinks auto_increment = 1")
       g.conn.commit()
       return jsonify({"result": True})


   if __name__ == "__main__":
  app.run(host='0.0.0.0')
```

MySQL Drink Database

```
-- MySQL dump 10.16  Distrib 10.1.23-MariaDB, for debian-linux-gnueabihf
(armv7l)
--
-- Host: localhost    Database: drinks
-- ------------------------------------------------------
-- Server version 10.1.23-MariaDB-9+deb9u1

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8mb4 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `drinks`
--

DROP TABLE IF EXISTS `drinks`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `drinks` (
  `drink_id` int(11) NOT NULL AUTO_INCREMENT,
  `drink_name` varchar(255) DEFAULT NULL,
  `drink_path` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`drink_id`)
) ENGINE=InnoDB AUTO_INCREMENT=12 DEFAULT CHARSET=utf8mb4;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `drinks`
--

LOCK TABLES `drinks` WRITE;
/*!40000 ALTER TABLE `drinks` DISABLE KEYS */;
INSERT INTO `drinks` VALUES (1,'Margarita','4,2,2,10,2,4,1,13,1'),(2,'Long
Island','7,2,1,4,1,11,1,6,1,12,1,8,1,10,1'),(3,'Jack &
Coke','2,6,2,12,2'),(4,'Sex on the
Beach','4,8,1.25,12,1.25,6,1,10,1'),(5,'Cape Cod','2,2,2.25,10,2'),(6,'7 &
7','2,6,2,10,2'),(7,'Jack & Coke
(double)','2,6,4,12,2'),(8,'Kamikaze','3,8,2,12,1,11,1'),(9,'Between the
Sheets','4,4,2,11,1,6,2,13,1'),(10,'Tequila
Sunrise','3,2,3,12,1,13,1'),(11,'Old Fashioned','1,6,3');
/*!40000 ALTER TABLE `drinks` ENABLE KEYS */;
UNLOCK TABLES;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
```

```
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2018-04-18  7:58:34
```

Graphical User Interface Code

```
from Tkinter import *
import Tkinter as tk
import tkMessageBox
import sys
import tkFont
import serial
import time
import os

f1 = "DrinkPic.gif"
f2 = "Marg.gif"
f3 = "seven.gif"
f4 = "BTS.gif"
f5 = "Cape.gif"
f6 = "JCoke.gif"
f7 = "Kami.gif"
f8 = "Ltea.gif"
f9 = "Ofash.gif"
f10 = "Sbeach.gif"
f11 = "Tsun.gif"
f12 = "apple.gif"
f13 = "orange.gif"
f14 = "coke.gif"
f15 = "pineapp.gif"

creds = 'tempfile.temp'
cred2 = 'tempfile2.temp'# This just sets the variable creds to
'tempfile.temp'
ser = serial.Serial('/dev/ttyACM0', 9600)
time.sleep(1)

global polyGUI

polyGUI= Tk()
polyGUI.title('POLYTELIA')
polyGUI.geometry("840x480+0+0")


mainfont = ('times', 60, 'bold', 'italic')
canvas = Canvas(polyGUI, width=840, height=480)
canvas.pack()
tk_img = PhotoImage(file = f1)
canvas.create_image(420,250, image=tk_img)
canvas.create_text(400,40, font=mainfont, text ='Polyteleia')

def settingsMenu():

    global smenu

    smenu = Toplevel()
    smenu.title('SETTING MENU')
    smenu.geometry("840x480+0+0")
    instrucFont=('times', 25, 'bold')
    instruction = Label(smenu, text='PLEASE SELECT A SETTINGS OPTION:\n')
    instruction.config(font= instrucFont)
```

```
    instruction.pack()

    def ShutDown():
        smenu.destroy()

    def NewUser():
        Signup()
        smenu.destroy()

    def NewPassword():
        ChangePass()
        smenu.destroy()

    def DeleteUser():
        UserWindow()
        smenu.destroy()

    Settingsfont=('times', 25, 'bold')

    newbutton = Button(smenu, text = 'New User',font = Settingsfont,
height=4, width=10, command = NewUser).place(x=30, y=80)
    passbutton = Button(smenu,text = 'Password\nReset', font = Settingsfont,
height=4, width=10, command = NewPassword).place(x=300, y=80)
    deletebutton = Button(smenu,text = 'Delete\nUser',font = Settingsfont,
height=4, width=10, command = DeleteUser).place(x=570, y=80)
    quitbutton = Button(smenu,text = 'System\nShutdown', font = Settingsfont,
bg='red', height=4, width=10, command = polyGUI.destroy). place(x=300,y=280)
    mainbutton = Button(smenu,text = 'Main\nMenu', height=2, width=10,
command = smenu.destroy).place(x=570,y=400)



    smenu.mainloop()


def kidsMenu():

    global kmenu

    kmenu = Toplevel()
    kmenu.title('KID MENU')
    kmenu.geometry("840x480+0+0")
    instrucFont=('times', 25, 'bold')
    instruction = Label(kmenu, text='PLEASE SELECT A BEVERAGE:\n')
    instruction.config(font= instrucFont)
    instruction.pack()

    def appleJuice():
         ser.isOpen()
         time.sleep(1)
         time.sleep(1)
         print ser.write("1,10,5\n")
         ser.close()
         print 'Apple Juice'
         kmenu.destroy()

    def orangeJuice():
```

```
          ser.isOpen()
          time.sleep(1)
          time.sleep(1)
          print ser.write("1,11,5\n")
          ser.close()
          print 'Orange Juice'
          kmenu.destroy()

     def coke():
          ser.isOpen()
          time.sleep(1)
          time.sleep(1)
          print ser.write("1,12,5\n")
          ser.close()
          print 'Coke'
          kmenu.destroy()

     def pineJuice():
          ser.isOpen()
          time.sleep(1)
          time.sleep(1)
          print ser.write("1,13,5\n")
          ser.close()
          print 'Pineapple Juice'
          kmenu.destroy()

     appPic = PhotoImage(file = f12)
     appbutton = Button(kmenu,text = 'Apple Juice',image = appPic,
justify=CENTER,compound= TOP, command = appleJuice).place(x = 30,y=180)

     orgPic = PhotoImage(file = f13)
     orgbutton = Button(kmenu,text = 'Orange Juice',image = orgPic,
justify=CENTER,compound= TOP, command = orangeJuice).place(x = 230,y=180)

     cokePic = PhotoImage(file = f14)
     cokebutton = Button(kmenu,text = 'Coca-Cola',image = cokePic,
justify=CENTER,compound= TOP, command = coke).place(x = 430,y=180)

     pinePic = PhotoImage(file = f15)
     pinebutton = Button(kmenu,text = 'Pineapple Juice',image = pinePic,
justify=CENTER,compound= TOP, command = pineJuice).place(x = 630,y=180)

     kmenu.mainloop()


def adultMenu():

     global amenu

     amenu = Toplevel()
     amenu.title('ADULT MENU')
     amenu.geometry("840x480+0+0")
     instrucFont=('times', 25, 'bold')
     instruction = Label(amenu, text='PLEASE SELECT A BEVERAGE:\n')
     instruction.config(font= instrucFont)
     instruction.pack()
```

```
def TequilaSunrise():
     ser.isOpen()
     time.sleep(1)
     time.sleep(1)
     print ser.write("3,2,3,12,1,13,1\n")
     ser.close()
     print 'Tequila Sunrise'
     amenu.destroy()

def Margarita():
     ser.isOpen()
     time.sleep(1)
     time.sleep(1)
     print ser.write("4,2,2,10,2,4,1,13,1\n")
     ser.close()
     print 'Margarita'
     amenu.destroy()

def LongIsland():
     ser.isOpen()
     time.sleep(1)
     time.sleep(1)
     print ser.write("7,2,1,4,1,11,1,6,1,12,1,8,1,10,1\n")
     ser.close()
     print 'Long Island'
     amenu.destroy()

def SexOnBeach():
     ser.isOpen()
     time.sleep(1)
     time.sleep(1)
     print ser.write("4,8,1.25,12,1.25,6,1,10,1\n")
     ser.close()
     print 'Sex on the beach'
     amenu.destroy()

def CapeCod():
     ser.isOpen()
     time.sleep(1)
     time.sleep(1)
     print ser.write("2,2,2.25,10,2\n")
     ser.close()
     print 'Cape Cod'
     amenu.destroy()

def SevenAndSeven():
     ser.isOpen()
     time.sleep(1)
     time.sleep(1)
     print ser.write("2,6,2,10,2\n")
     ser.close()
     print '7 and 7'
     amenu.destroy()

def Kamikaze():
     ser.isOpen()
```

```
     time.sleep(1)
     time.sleep(1)
     print ser.write("3,8,2,12,1,11,1\n")
     ser.close()
     print 'Kamikaze'
     amenu.destroy()


def BwtSheets():
     ser.isOpen()
     time.sleep(1)
     time.sleep(1)
     print ser.write("4,4,2,11,1,6,2,13,1\n")
     ser.close()
     print 'Between the Sheets'
     amenu.destroy()

def JackCoke():

    shot()
    shotCheck = shotVal.get()

    if shotCheck == 1:
        ser.isOpen()
        time.sleep(1)
        time.sleep(1)
        print ser.write("2,6,2,12,2\n")
        ser.close()
        print'Single Jack and Coke'

    else:
        ser.isOpen()
        time.sleep(1)
        time.sleep(1)
        print ser.write("2,6,4,12,2\n")
        ser.close()
        print'Double Jack and Coke'

    Shot.destroy()
    amenu.destroy()

def OldFashion():

    shot()
    shotCheck = shotVal.get()

    if shotCheck == 1:
        ser.isOpen()
        time.sleep(1)
        time.sleep(1)
        print ser.write("1,6,3\n")
        ser.close()
        print'1shotwhiskey'

    else:
        ser.isOpen()
        time.sleep(1)
```

```
            time.sleep(1)
            print ser.write("1,6,4.5\n")
            ser.close()
            print'2shotwhiskey'

        Shot.destroy()
        amenu.destroy()



    can = Canvas(amenu, width = 800, height =700)
    menuframe = Frame(can, width = 825, height =700, bd =2, bg ="white",
relief =SUNKEN)
    scrollbar = Scrollbar(amenu, orient="vertical", command=can.yview)
    can.configure(yscrollcommand=scrollbar.set)
    scrollbar.pack(side="right", fill= "y")
    can.pack()
    can.create_window((400,400), window=menuframe, tags="menuframe")
    task=can.configure(scrollregion=can.bbox("all"))
    menuframe.bind(task)


    Marg_pic = PhotoImage(file = f2)
    Margbutton = Button(menuframe,text = 'Margarita',image = Marg_pic,
justify=CENTER,compound= TOP, command = Margarita).place(x = 30,y=10)

    sevenpic= PhotoImage(file = f3)
    sevenbutton = Button(menuframe,text = '7 and 7',image = sevenpic,
justify=CENTER,compound= TOP, command = SevenAndSeven).place(x = 230,y=10)

    BTSpic = PhotoImage(file = f4)
    BTSbutton = Button(menuframe,text = 'Between the Sheets',image = BTSpic,
justify=CENTER,compound= TOP, command = BwtSheets).place(x = 430,y=10)

    Capepic = PhotoImage(file = f5)
    Capebutton = Button(menuframe,text = 'Cape Cod',image = Capepic,
justify=CENTER,compound= TOP, command = CapeCod).place(x = 630,y=10)

    Jcokepic = PhotoImage(file = f6)
    Jcokebutton = Button(menuframe,text = 'Jack and Coke',image = Jcokepic,
justify=CENTER,compound= TOP, command = JackCoke).place(x = 30,y=220)

    Kamipic = PhotoImage(file = f7)
    Kamibutton = Button(menuframe,text = 'Kamikaze',image = Kamipic,
justify=CENTER,compound= TOP, command = Kamikaze).place(x = 230,y=220)

    Ltea_pic = PhotoImage(file = f8)
    Lteabutton = Button(menuframe,text = 'Long Island Iced Tea',image =
Ltea_pic, justify=CENTER,compound= TOP, command = LongIsland).place(x =
430,y=220)

    Ofash_pic = PhotoImage(file = f9)
    Ofashbutton = Button(menuframe,text = 'Old Fashion',image = Ofash_pic,
justify=CENTER,compound= TOP, command = OldFashion).place(x = 630,y=220)

    SBpic = PhotoImage(file = f10)
```

```
    SBbutton = Button(menuframe,text = 'Sex on the Beach',image = SBpic,
justify=CENTER,compound= TOP, command = SexOnBeach).place(x = 30,y=430)

    Tsun_pic = PhotoImage(file = f11)
    Tsunbutton = Button(menuframe,text = 'Tequila Sunrise',image = Tsun_pic,
justify=CENTER,compound= TOP, command = TequilaSunrise).place(x = 230,y=430)

    amenu.mainloop()

def Signup():
    global pwordE
    global nameE
    global roots

    roots = Tk()
    roots.title('Signup')
    intruction = Label(roots, text='Please Enter new Credidentials\n')
    intruction.grid(row=0, column=0, sticky=E)

    nameL = Label(roots, text='New Username: ')
    pwordL = Label(roots, text='New Password: ')
    nameL.grid(row=1, column=0, sticky=W)
    pwordL.grid(row=2, column=0, sticky=W)

    nameE = Entry(roots)
    pwordE = Entry(roots, show='*')
    nameE.grid(row=1, column=1)
    pwordE.grid(row=2, column=1)

    signupButton = Button(roots, text='Signup', command= FSSignup)
    signupButton.grid(columnspan=2, sticky=W)
    roots.mainloop()

def FSSignup():
    with open(cred2, 'w') as f:
        f.write(nameE.get())
        f.write('\n')
        f.write(pwordE.get())
        f.close()

    roots.destroy()

def Login():
    global nameEL
    global pwordEL
    global rootA

    rootA = Tk()
    rootA.title('Login')

    intruction = Label(rootA, text='Please Login\n')
    intruction.grid(sticky=E)

    nameL = Label(rootA, text='Username: ')
    pwordL = Label(rootA, text='Password: ')
    nameL.grid(row=1, sticky=W)
    pwordL.grid(row=2, sticky=W)
```

```
    nameEL = Entry(rootA)
    pwordEL = Entry(rootA, show='*')
    nameEL.grid(row=1, column=1)
    pwordEL.grid(row=2, column=1)

    loginB = Button(rootA, text='Login', command=CheckLogin)
    loginB.grid(columnspan=2, sticky=W)

    canCel = Button(rootA, text='Cancel', fg='red', command=Cancel)
    canCel.grid(columnspan=2, sticky=W)
    rootA.mainloop()

def Cancel():
    rootA.quit()
    rootA.destroy()


def CheckLogin():
    with open(creds) as f:
        data = f.readlines()
        uname = data[0].rstrip()
        pword = data[1].rstrip()

    if nameEL.get() == uname and pwordEL.get() == pword:
        rootA.quit()
        rootA.destroy()
        adultMenu()

    else:
        CheckLogin2()

def CheckLogin2():
    with open(cred2) as f:
        data = f.readlines()
        uname = data[0].rstrip()
        pword = data[1].rstrip()

    if nameEL.get() == uname and pwordEL.get() == pword:
        rootA.quit()
        rootA.destroy()
        adultMenu()

    else:
        r = Tk()
        r.geometry('150x50')
        rlbl = Label(r, text='\nFailed Login')
        rlbl.pack()
        r.mainloop()
        Login()

def UserWindow():

    global nameEL2
    global pwordEL2
    global rootD
```

```
    rootD = Tk()
    rootD.title('Delete User')

    intruction = Label(rootD, text='Enter User for Deletion:\n')
    intruction.grid(sticky=E)

    nameL = Label(rootD, text='Username: ')
    pwordL = Label(rootD, text='Password: ')
    nameL.grid(row=1, sticky=W)
    pwordL.grid(row=2, sticky=W)

    nameEL2 = Entry(rootD)
    pwordEL2 = Entry(rootD, show='*')
    nameEL2.grid(row=1, column=1)
    pwordEL2.grid(row=2, column=1)

    loginB = Button(rootD, text='Delete', command=DelUser)
    loginB.grid(columnspan=2, sticky=W)

    canCel = Button(rootD, text='Cancel', fg='red', command=rootD.destroy)
    canCel.grid(columnspan=2, sticky=W)

    rootD.mainloop()


def DelUser():

    os.remove(cred2)
    rootD.destroy()
    tkMessageBox.showwarning("Warning","User has been deleted")

def ChangePass():

    global nameEL3
    global pwordEL3
    global rootP

    rootP = Tk()
    rootP.title('New Password')

    intruction = Label(rootP, text='Enter Credentials\n')
    intruction.grid(sticky=E)

    nameL = Label(rootP, text='Username: ')
    pwordL = Label(rootP, text='Password: ')
    nameL.grid(row=1, sticky=W)
    pwordL.grid(row=2, sticky=W)

    nameEL3 = Entry(rootP)
    pwordEL3 = Entry(rootP, show='*')
    nameEL3.grid(row=1, column=1)
    pwordEL3.grid(row=2, column=1)

    loginB = Button(rootP, text='Enter', command=updatePass)
    loginB.grid(columnspan=2, sticky=W)

    canCel = Button(rootP, text='Cancel', fg='red', command=rootP.destroy)
```

```
    canCel.grid(columnspan=2, sticky=W)

    rootP.mainloop()

def updatePass():
    global pword1
    global pword2
    global rootP1

    def check():

        if pword1.get() == pword2.get():
            with open(cred2, 'w') as f:
                f.write(nameEL3.get())
                f.write('\n')
                f.write(pword1.get())
                f.close()
                rootP.destroy()
                rootP1.destroy()
        else:
            ChangePass()
            rootP1.quit()


    rootP1 = Tk()
    rootP1.title('New Password')

    intruction = Label(rootP1, text='Enter Credentials\n')
    intruction.grid(sticky=E)

    pword1 = Label(rootP1, text='Password: ')
    pword2 = Label(rootP1, text='Password: ')
    pword1.grid(row=1, sticky=W)
    pword2.grid(row=2, sticky=W)

    pword1 = Entry(rootP1,show='*')
    pword2 = Entry(rootP1, show='*')
    pword1.grid(row=1, column=1)
    pword2.grid(row=2, column=1)

    loginB = Button(rootP1, text='Enter', command=check)
    loginB.grid(columnspan=2, sticky=W)

    canCel = Button(rootP1, text='Cancel', fg='red', command=rootP1.destroy)
    canCel.grid(columnspan=2, sticky=W)


    rootP1.mainloop()

def shot():

    global Shot
    global shotVal

    Shot = Toplevel()
    Shot.geometry("300x200+200+200")
    shotFont=('times', 12, 'bold')
```

```
    shotlabel = Label(Shot, text='Please select number of shots:\n')
    shotlabel.config(font= shotFont)
    shotlabel.pack()

    shotVal=IntVar()
    shotVal.set(0)
    radio1 = Radiobutton(Shot,text = 'Single\nShot',value = 1,
variable=shotVal, command= Shot.quit).pack(side='top')
    radio2 = Radiobutton(Shot,text = 'Double\nShot',value = 2,
variable=shotVal, command= Shot.quit).pack(side='bottom')

    Shot.mainloop()

labelfont=('times', 25, 'bold')
settingsbutton = Button(text = 'Settings',font = labelfont, bg='red',
height=5, width=10, command = settingsMenu).place(x=30, y=210)
kidbutton = Button(text = 'Kid''s Menu', font = labelfont, bg='blue',
height=5, width=10, command = kidsMenu).place(x=300, y=210)
adultbutton = Button(text = 'Adult Menu',font = labelfont, bg='yellow',
height=5, width=10, command = Login).place(x=570, y=210)

polyGUI.mainloop()
```

Bibliography

[1] “Alcoholic Beverage Market Overview In The United States.” *Park Street Imports*, www.parkstreet.com/alcoholic-beverage-market-overview/.

[2] *Arduino - Software*, www.arduino.cc/en/Main/Software.

[3] “Chapter 1 General Information.” *MySQL*, dev.mysql.com/doc/refman/5.7/en/introduction.html.

[4] “Connect Raspberry Pi and Arduino with Serial USB Cable.” *Oscar Liang*, 23 Mar. 2018, oscarliang.com/connect-raspberry-pi-and-arduino-usb-cable/.

[5] “Help Guides and Resources - How to Use Raspberry Pi.” *Raspberry Pi*, www.raspberrypi.org/help/.

[6] “24.1. Tkinter - Python Interface to Tcl/Tk¶.” *24.1. Tkinter - Python Interface to Tcl/Tk - Python 2.7.15rc1 Documentation*, Python , docs.python.org/2/library/tkinter.html.