

**Senior Design Project**

**Design of An Automated Alcohol Dispensing System**

ENGE476 Senior Design Project I

Department of Engineering and Aviation Sciences

University of Maryland, Eastern Shore

Nathan Bane

Chris Blanks

Ryan Valente

Project Advisor, Dr. Lei Zhang

Submitted

Nov. 17, 2018

List of Contents

[List of Contents 2](#_Toc525994184)

[List of Figures 4](#_Toc525994185)

[List of Tables 5](#_Toc525994186)

[Abstract 6](#_Toc525994187)

[1. Introduction 7](#_Toc525994188)

[1.1 Backgound/Motivation 7](#_Toc525994189)

[1.2 Objective 8](#_Toc525994190)

[1.3 Design Requirements 8](#_Toc525994191)

[1.4 Design Constraints 8](#_Toc525994192)

[1.5 Design Methods 8](#_Toc525994193)

[2. Project Description 9](#_Toc525994194)

[2.1 System Description 9](#_Toc525994195)

[2.2 System Diagram 9](#_Toc525994196)

[2.3 System Functions 11](#_Toc525994197)

[3. Implementation Plan 12](#_Toc525994198)

[3.1 Tasks 12](#_Toc525994199)

[3.2 Team Organization 14](#_Toc525994200)

[3.2.1 Responsibility of Nathan. 14](#_Toc525994201)

[3.2.2 Responsibility of Chris. 14](#_Toc525994202)

[3.3 Timeline/Milestones/Delivery Plan 14](#_Toc525994203)

[4. Implementation 15](#_Toc525994204)

[4.1 Implementation of Task 1. 15](#_Toc525994205)

[4.2 Implementation of Task 1. 15](#_Toc525994206)

[5. Conclusion. 16](#_Toc525994207)

[Acknowledgement 17](#_Toc525994208)

[Appendix 18](#_Toc525994209)

[A. Component Specs 18](#_Toc525994210)

[1. Specs of the Main Computer 18](#_Toc525994211)

[2. Specs of the Embedded Board 18](#_Toc525994212)

[B. Source Code. 18](#_Toc525994213)

[1. Source Code of the Python Graphic User Interface 18](#_Toc525994214)

[2. Source Code of the Android Application 18](#_Toc525994215)

[References 19](#_Toc525994216)

List of Figures

[Fig. 1. Embedded System Diagram: Main Computer, Hardware Controller 11](#_Toc529379188)

[Fig. 2. Main Computer Application Diagram 11](#_Toc529379189)

[Fig. 3. Phone Application Diagram 12](#_Toc529379190)

[Fig. 4. Flow chart of System While In Customer Mode 12](#_Toc529379191)

[Fig. 5. MVC Pattern 16](#_Toc529379192)

[Fig. 6. Class Diagram 16](#_Toc529379193)

[Fig. 7. Parent and Child Directories 17](#_Toc529379194)

[Fig. 8. Example of the Text File Format of Each Drink Profile 17](#_Toc529379195)

List of Tables

[Table 1. Project Timeline and Delivery Plan 14](#_Toc529383115)

[Table 2. Project Timeline and Delivery Plan (Continued) 15](#_Toc529383116)

Abstract

In Progress.

1. Introduction

Over the course of the project the team will detail the development of an automated drink making machine that will alleviate wait times in bars, allow users to pick complicated drinks, and facilitate a good buying experience for the customers.

## Backgound/Motivation

In most popular bars, there is always the issue of long lines of customers and overworked bartenders. Why does this usually occur? The ratio of bartenders to customers is usually very small, so there is more demand than supply. This issue can cause a few problems for the bartenders and customers. If the bartenders rush to meet the demand of the customers then that will liken the chance of errors being made in the drink making process, which will lower the quality of the drinks and cause customers to become unsatisfied with the service. If the bartenders are too slow to meet the demand, then some customers will become irritated due to the long wait time. When wait times get too long, customers tend to migrate to other bars that are not as busy. This is very disadvantageous for the bar owner and bartender because that means less revenue. Limiting the number of customers in one period or space is an option, but that does not maximize on the potential profit.

One solution that some bar owners take is to just hire more staff members, but this is not always feasible due to limited budgets and lack of space. Additionally, you cannot always predict when a large wave of people will come to the bar, so it is better to have a more flexible solution that takes up a very small amount of space, does not cost thousands of dollars, and can be reliable at any moment. The perfect solution to help bar owners and bartenders is an automated machine that meets the previously listed criteria. This machine would be able to work at any moment of the day and make bartenders more productive.

After doing some market research on the usefulness of an automated drink making machine, the machine must be easily customizable and intuitive to use. When the machine is easily customizable, a bar owner can make more use of the machine and cater to multiple groups of people. For example, if the bar owner can add or exchange the drinks in the machine easily then the owner can meet the needs of the customers a lot faster and enhance their experiences. An intuitive interface is a very important feature of any product. When it is simple to use, it will be used a lot more than if it was convoluted. In addition, if the bar employees have a hard time trying to figure out how to use the machine then that makes them less productive, which means less revenue. Ideally, any bar owner would want a machine that would augment the abilities of their workers.

A couple other considerations for an automated drinking machine are the payment process and the issue of people that have had too many drinks. Traditionally, most customers to bars will either pay upfront for their drinks or will ask the bartender to open a tab until they are done. Sometimes customers drink too much, so they are cut off from buying any more drinks. This is a task that is usually left to the bartender’s discretion. An automated machine can help with these two common tasks. The automated machine could handle cash, credit cards, and electronic payments (e.g. PayPal, Venmo, Bitcoin). This could potentially allow more money to be spent at the bar. With the automated machine handling drinks, the drink making process would be objective and customers would get exactly what they bought. The issue of preventing customers from drinking too much is also an achievable task for an automated machine. The machine could incorporate a system for detecting when a customer is past an acceptable level of drunkenness. This could be done with the use of a facial recognition camera that monitors the facial expressions of recurring customers over the night and decides whether they should have another drink within a certain time frame. Another option is using an infrared camera to do a similar job as the facial recognition camera. Regardless of the method, the maker of an automated drink making machine must take these two issues into consideration.

There a few solutions on the market already for automating the drink making process. The first solution used the layout of a 2-D plotter to design their automated drink making machine. The alcoholic beverages are fastened in their own slots and with their bottoms upward. The alcohol is dispensed by utilizing 25 ml Beaumont Taps. A servo actuates these taps and dispenses the alcohol into a cup that is attached to a linear actuator. The cup is then moved to the next drink dispensing location for the next ingredient for a mixture. Once the mixture has been finished, the patron picks up the cup from the machine. There are also lights that are integrated into the design that make their drink making process more entertaining. The user interface is just an array of buttons and a Liquid Crystal Display (LCD) screen.

The second solution utilizes two industrial robotic arms to mix a wide range of drinks. The robotic arms are located on a platform about the height of a counter, and the drinks are suspended above the robot platform. It uses the common utensils that a standard bartender uses. This allows the robot to make any drink that a bartender can make. The robot arms are also capable muddling, stirring, shaking, and straining actions, which are common for drink making. This is a very extendable design because the robots are dexterous enough to perform many actions, so this solution can make a lot of drinks. It can make two drinks per minute, which yields around 1,000 drinks per day.

Both solutions can create alcoholic drinks and allow patrons easy access to drinks. However, the first solution has a slow drink making process. Each pour the machine makes is a pre-calibrated pour that cannot be changed. As a result, it can only mix imprecise volumes. In addition, the machine’s unwieldy size is not suitable for most bar applications. The electronics and mechanism are open to the patrons allowing them to potentially damage the machine. Another con is that it could expose drinks to impurities before the customer gets the drink. The machine also doesn’t dispense ice or other ingredients, so only basic drinks can be made. The second solution would be very expensive and too large for most bar owners. It also requires a lot of overhead to operate. The industrial robot arms are heavy pieces of equipment which will require special structures to support. It will also have high power requirements necessitating the need for a specific electric service that is not always available. These specific needs require a huge investment to install. This puts it out of reach for most bar owners.

The designing and building of an efficient automated drink making machine is no easy task due to all the considerations to keep in mind. It requires keeping the bar owner, bartender, and customer happy with the product. It requires a flexible and extensible design, so that it can be appealing to large groups of people. It is a suitable engineering challenge to undertake.

## Objective

Design an automated drink making machine that will offer a more economical solution to reduce wait times for drinks, create drinks, and reduce the human error in the drink making process.

## Design Requirements

1. The machine will fit in a 3.5’ x 2.5’ x 3’ (L x W x H) space.
2. The dry weight, the machine without ice or fluids, of the machine will be less than 50 lbs.
3. It will be able to serve at least 10 different drinks due to the combinations of 4 liquors, 1 liqueur, and 7 flavoring additives.
4. The machine will feature an easy to use graphical user interface allowing the patrons to select various drinks for purchase.
5. The machine will be able to be powered from 120/220 V outlets.
6. The machine will check licenses for the legal drinking age if customers are directly interacting with the machine.
7. The phone application will allow store managers to monitor the machine in real-time and view drink analytics.
8. This machine must securely store profits from customers.

## Design Constraints

1. The machine must conform to machine food and safety requirements.
2. The touchscreen interface must be close to eye level of users.

## Design Methods

The first step is to design the board for the embedded system section. The second step is to design a computer application for the main computer. The third step is to write code to interface all the peripherals with the main computer and embedded board. The fourth step is to design a phone application that can pull system data from the main computer. The fifth step is to identify a power supply that will support the whole machine. The sixth step is to implement image recognition using the camera and main computer. The final step is to use a Field Programmable Gate Array to create extra features for the machine. After these steps are completed, our project will be done.

1. Project Description

## System Description

This system is composed of three main systems: the embedded electronics, the main computer application, and the phone application.

The embedded electronics can be broken down into two sections: the hardware controller board and the main computer. The hardware controller board will use a microcontroller to control fluid flow throughout the system via controlling valves, dc brushed motors, and peristaltic pumps. An additional motor controller will be used to dispense ice from the ice container. An onboard sensor module will be used to gather data regarding the operation of the machine. The main computer will host the touchscreen application that customers will use in order to select and purchase drinks from the system. The main computer will also interface with the payment collector, Field Programmable Gate Array (FPGA), and camera. The payment collector will send signals to the main computer when payment has been received. The FPGA will control extra features (e.g. LED light show, security). The camera will be used to detect whether a customer is too inebriated.

The main computer application will control the states of the system (e.g. wait, dispensing, cleaning) through a series of commands sent to peripheral devices. It will be an application made with the Python 3 programming language and its supporting libraries. The app will have a graphical user interface (GUI) that will contain buttons, images, and dropdown menus. The customer will interact with the application through the touchscreen. Information regarding the inventory, purchases/sales, and customers will be stored in the local file system of the main computer.

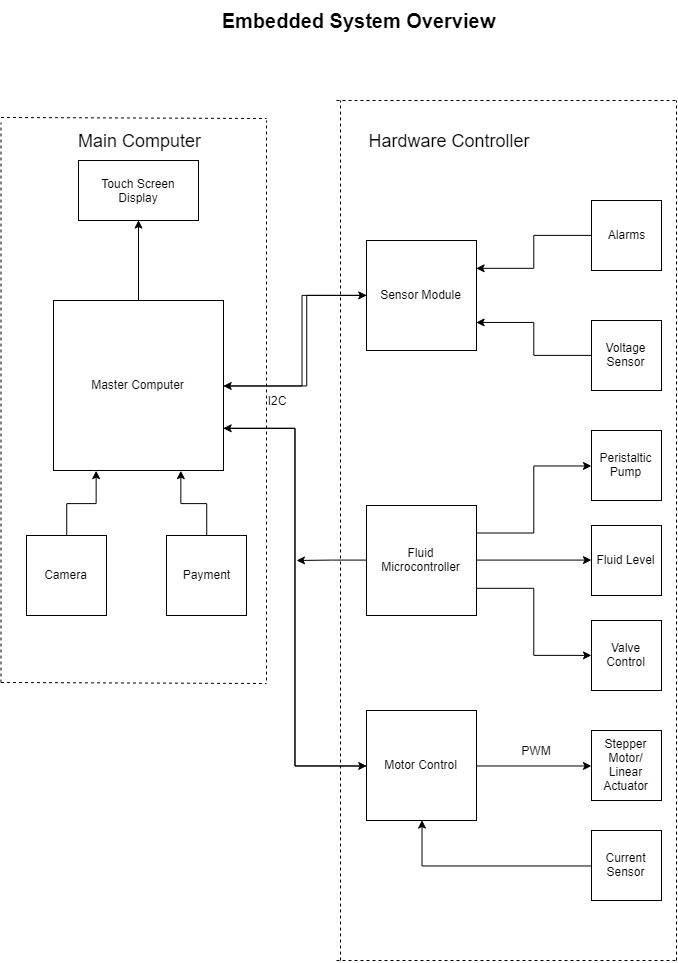
The phone application will be used for remote monitoring and configuring of the system. The app will be created in Android Studio using the Java programming language and will be available for any android phone. The application will consist of a main menu, buttons, dropdowns, graphical displays, and images. In order for the app to acquire and display data about the whole system, it will have to use the file transfer protocol (FTP) to gain access to the file system of the main computer. In addition, the app will be able to change system configuration settings by editing a special file in the main computer.

## System Diagram

### Hardware:

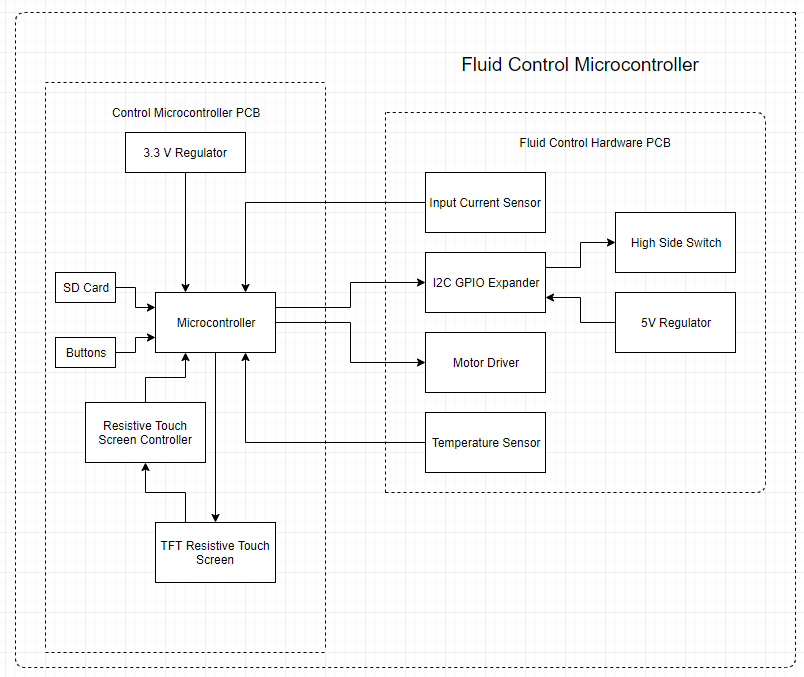
The **Embedded System Overview** is divided into two sections the **Main Computer** and the **Hardware Controller**. The main computer provides and interface between the customer through the use of a touchscreen and the hardware such as pumps, valves, and sensors. The main computer is a Raspberry Pi 3 a 1.2 GHz Linux capable computer. The customers order is placed on the tough screen and executed by sending I2C Commands to the hardware controller. It enables the two sections to accomplish different tasks at the same time.

The hardware controller completes the interface between the main computer through the use of an STM32 microcontroller and the hardware. The onboard microcontroller receives its commands through the I2C Bus. Through the use of clever programming will command its various subassemblies to actuate the motors, valves, and other hardware. The hardware controller has three main sections the **Sensor Module** which senses important data such as the security of the system through tamper alarms, the **Fluid Microcontroller** controls flow of fluid through the system, and the **Motor Control** section will oversee any additional motor control that is needed.



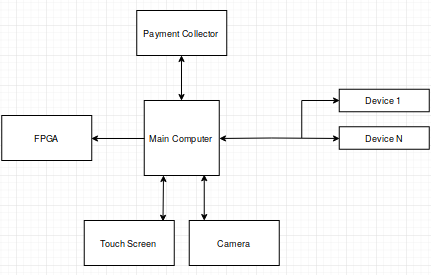
1. Embedded System Diagram: Main Computer, Hardware Controller

The **Fluid Microcontroller** controls the flow of fluid through out the system. It has two main methods for receiving its commands is through the I2C Bus or through the onboard touchscreen. This enables the system to opperate as a stand-alone unit or in direct command mode. The command will be excuted through the gpio pins on the microcontroller to the various subassembles such as the high-side switch and motor driver. The fluid control hardware pcb has some built in support circuitry to enable smooth operation of the pcb such as 3.3V to 5V translation hardware to enable the STM32 microcontroller to communicate with the hardware.

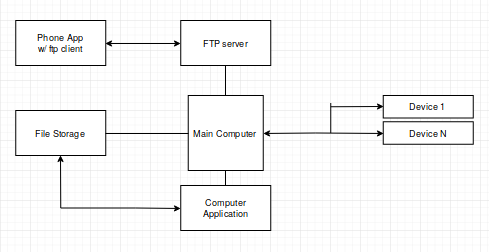


1. Fluid Control Microcontroller

### Software

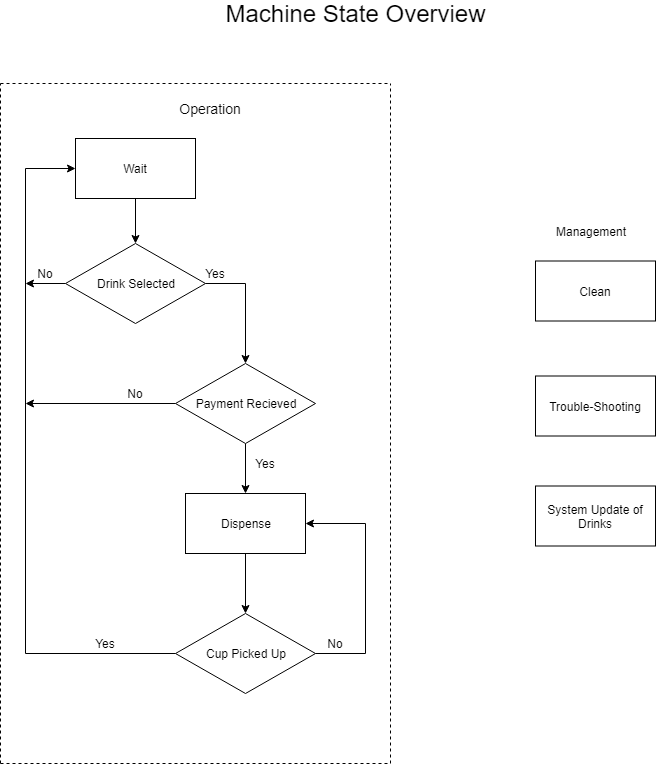


1. Main Computer Application Diagram



1. Phone Application Diagram

## System Functions



1. Flow chart of System While In Customer Mode
2. Implementation Plan
3. Design sub-assemblies for the embedded board
   1. Design high side switch to switch the solenoid valve on and off
   2. Design a stepper motor driver and supporting circuitry
   3. Design a 3.3 V regulator
4. Select/Integrate Microcontroller

Subtask 1. Select microcontroller and develop supporting circuitry for microcontroller.

Subtask 2. Select and design supporting circuitry for I2C EEPROM

Subtask 3. Develop circuitry to interface with a resistive touch screen

Subtask 4. Interface switches with microcontroller to manipulate modes of microcontroller

Subtask 5. Layout printed circuit board

Subtask 6. Assemble components onto printed circuit board

Subtask 7. Code Microcontroller.

Subtask A. Stand-Alone Mode: Code the microcontroller to interface with a touch screen to allow for direct command of the microcontroller.

Subtask B. Slave Mode: Code the microcontroller to receive commands through the serial data line

1. Develop a communication protocol between a shared I2C EEPROM, the main computer, and the embedded development board.
2. Create a main computer application
   1. Design a GUI using Python’s Tkinter Library that the customers will use
   2. Develop an efficient way to organize monitoring/drink information in the file storage system
   3. Add in capabilities for serial communication between the main computer and peripheral devices.
   4. Interface the main computer with the embedded boards and peripherals.
3. Create a phone application
   1. Design the user interface of the phone app with Android Studio’s SDK
   2. Allow the phone app to access the system information stored on the main computer and acquire relevant data for use within the app
   3. Allow the phone app to access the system information stored on the main computer and acquire relevant data for use within the app
   4. Allow the user to update system information (e.g. drinks currently stored)
4. Add extra capabilities and entertaining features
   1. Design an LED lightshow to catch the customer’s attention
   2. Use an ultrasonic sensor to trigger the entertaining features whenever a potential customer is close
   3. Build an alarm system to detect forced entry into the internals of the system
   4. Incorporate facial recognition into the main application.

## Team Organization

### Responsibility of Nathan.

Task 1, Task 2, Task 3

### Responsibility of Chris.

Task 3, Task 4, Task 5, Task 6

## Timeline/Milestones/Delivery Plan

1. Project Timeline and Delivery Plan

|  |  |  |  |
| --- | --- | --- | --- |
| **Time** | **Task** | **Comments** | **Responsible Personnel** |
| Fall 2018 Semester | | |  |
| Week 1-4 | N/A | Project proposal and planning | Nathan Bane |
| Week 4-  Week 7 | Task 1 (subtask 1.1,1.2) | Design and draw up a schematic for a high-side switch. Develop circuitry to control a brushed DC Motor . |
| Week 8-  Week 10 | Task 1(subtask 1.3-1.5) | Design and layout circuitry to support an I2C Thermometer, Current Sense Circuitry, and I2C GPIO Expander |
| Week 11- Week 13 | Task 1(subtask 1.6-1.7) | Layout the components on the Printed Circuit Board and order Printed Circuit Board with parts |
| Week 13 - End of Semester | Task 1(subtask 1.8-1.9) | Solder components onto the printed circuit board and develop code to test the printed circuit board |
| Spring 2019 Semester | | |  |
| Week 1-4 | Task 2 (subtask 2.1-2.4), Task 3 | Choose a suitable microcontroller and develop supporting circuitry.  Develop communication protocol | Nathan Bane & Christopher Blanks |
| Week 5-  Week 8 | Task 2(subtask2.5-2.6) | Layout printed circuit board, order components , and assemble components onto printed circuit board | Nathan Bane |
| Week 9  -End of the semester | Task 2(subtask 2.4 ) | Develop Code for Embedded |

1. Project Timeline and Delivery Plan (Continued)

|  |  |  |  |
| --- | --- | --- | --- |
| **Time** | **Task** | **Comments** | **Responsible Personnel** |
| Fall 2018 Semester | | |  |
| Week 1-5 | N/A | Project proposal and planning | Chris Blanks |
| Week 6-  Week 11 | Task 4  (subtask 4.1,4.2, 4.3) | Design and create the main application. Test the inputs and outputs of the system, so that it will interface correctly with the embedded boards and other peripherals. |
| Week 12-  End of the semester | Task 5  (subtask 5.1, 5.2, 5.3) | Design and create the phone application. |
| Spring 2019 Semester | | |
| Week 1-6 | Task 4  (subtask 4.4 and 4.5 ) | Finish interfacing and testing the main computer/application with the other systems. Develop an algorithm for checking driver licenses. |
| Week 7-  Week 11 | Task 6  (subtask 6.1 ,6.2, 6.3) | Design and implement the LED lightshow and alarm system. Incorporate facial recognition into the main application. |
| Week 11  -End of the semester | Task 5(subtask 5.4 ) | Add extra features to the phone application. |

1. Implementation

## Implementation of Task 1.

### Design a Valve and Pump Control Board

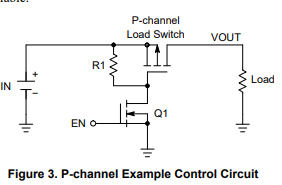
The purpose of the valve and pump control board is to facilitate control of the hardware necessary to control the flow of fluid through-out the system. It receives commands from the control microcontroller pcb through the I2C Bus and translates those commands into actions through various subassemblies. There are 5 main sub-assemblies: high-side switch, dc brushed motor driver, I2C GPIO expander, I2C thermometer, and input power current sensor. The high-side switch controls the valves directing the flow of fluid

### High-Side Switch

There are two main types of load switches that are used in circuitry: the high-side switch which utilizes a PMOS transistor and the low-side switch which utilizes the NMOS transistor. These switches enable loads to be switch on and off thus conserving power and controlling the flow of power through the system. It is often used in battery powered systems to turn off less used sections of the system to conserve power and then turn power on when that section is needed.

The high side switch utilizes a PMOS to switch the load on and off. The PMOS has some advantages and disadvantages over the NMOS. The advantage of the PMOS is that the control circuitry is much simpler than the NMOS. In order to turn the PMOS on the voltage on the source pin must be greater than the threshold voltage and the gate voltage. This allows the control circuitry to use the input voltage to control the high-side switch. The disadvantage to the high-side switch is that PMOS generally suffer from higher power loses due to RDS values due to the construction of the MOSFET.

In order to turn on and off the high-side switch the gate pin on the PMOS transistor is connected to ground through a logic level NMOS transistor. The utilization of the logic level NMOS transistor is to enable a microcontroller to operate the high-side switch. The NMOS transistor has a current limiting resistor to prevent the NMOS transistor from shorting the power rail to ground.

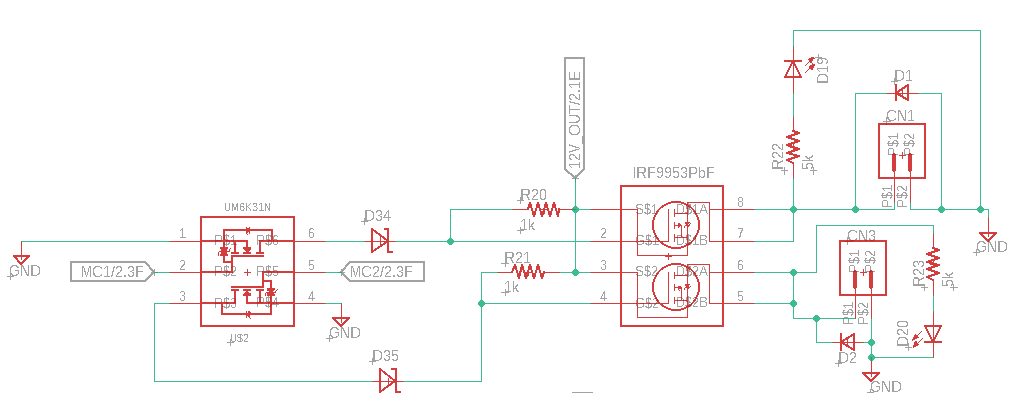


1. High-Side Switch

The requirements for our high-side switch was determined by the solenoid valve chosen to control the flow of liquid through the system. We chose values 150% of current requirements which means that all parts can handle at least .75 amps and 24 V. The requirements allow the circuit to support manufacturing imperfection in the resistance of the coil of the solenoid and any random current spikes that could potentially happen.

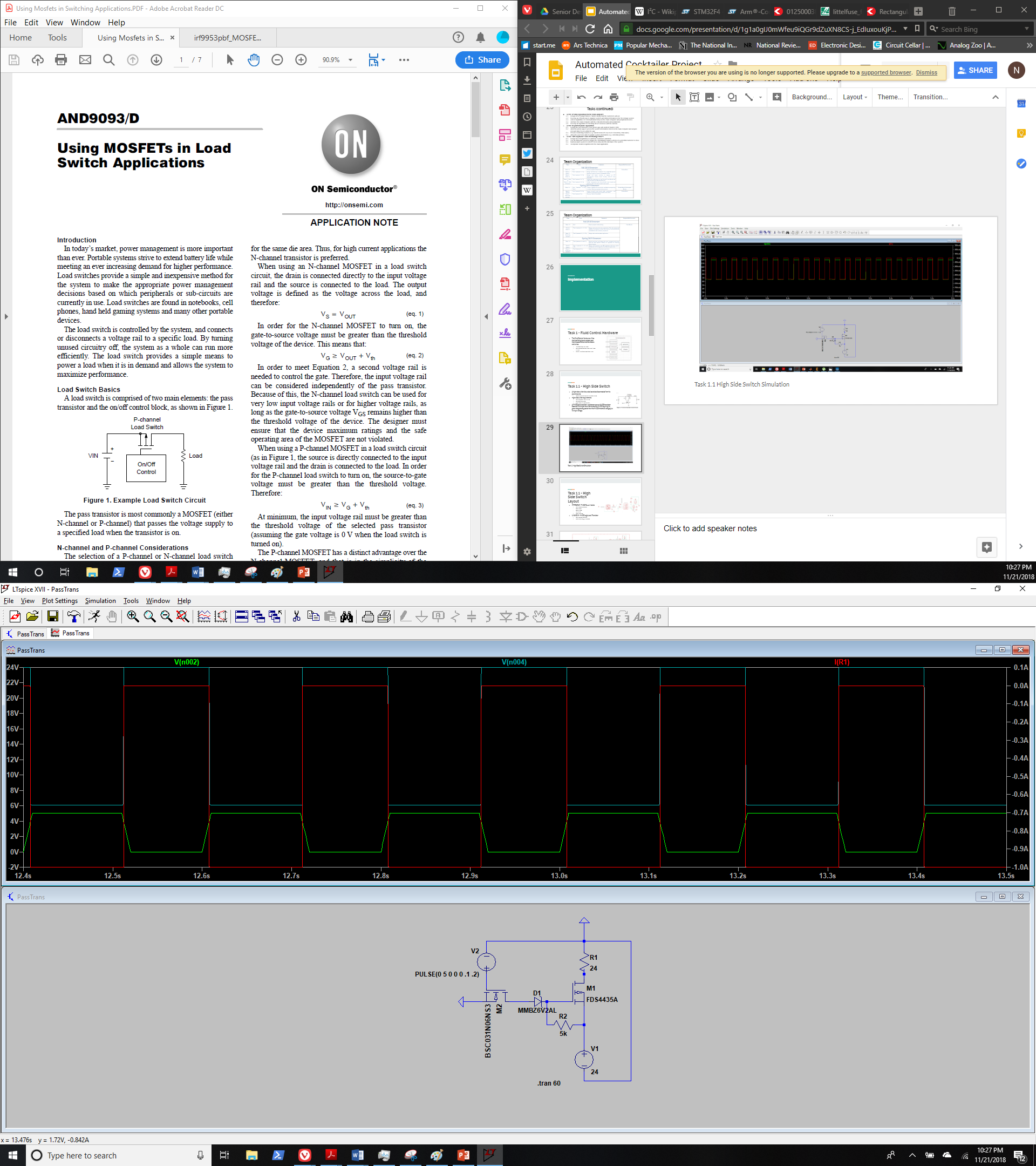
The load switch chosen for this application is a dual PMOS package is the International Rectifier IRF9953PBF that can handle -30 VDSS, 1.8 A, -20 VGS, and -1 VTH. The dual PMOS package was chosen in order to reduce the number of components needed to solder and save on board area. It will reduce the number from 16 individual pass transistor into 8 dual package transistors.

When the IRF9953PBF was originally chosen the voltage requirements were only 12 V so the -20 volts between the gate and the source was not a problem. However, when it was determined that the voltage requirements would need to be raised to 24 V in order to reduce the current flowing through the board the max VGS became a problem. In order to prevent the voltage from exceeding the max VGS it was determined that a zener diode would need to placed in series with the control transistor. The zener diode was reversed biased to raise the voltage from ground to the avalanche voltage of the zener diode. The avalanche voltage chosen for this project was 6.2 V because it will protect VGS when it is 24 V and allow the pass transistor to operate when it is 12 V.



1. Schematic for High-Side Switch

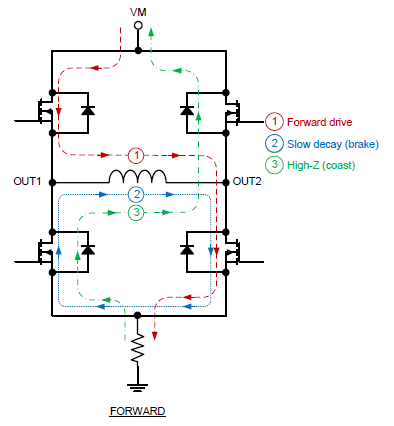
In order to confirm the design of the circuit it was simulated in LTSpice an open source circuit simulation program. LTSpice did not have the components that we needed to simulate so similar components were chosen to substitute those components. The circuit was created in LTSpice and simulated to determine the overall function of circuit. The simulation node V(n004) is VGS of the MOSFET which can be seen does not exceed a difference of -20 V. The logic voltage V(n002) switches the pass transistor and current flows through I(R1) which is the load.



1. High-Side Switch Simulation

### Motor Driver

A motor driver is needed to control the DC brushed motor peristaltic pump. It will control the volume and flow rate of the fluid going to the dispensing location. The basic theory behind a motor driver a full h bridge. A full h bridge motor driver consists of 2 NMOS transistors which are in series connecting VCC to GND and an additional 2 NMOS transistors which are is series connecting VCC to Gnd. One lead of the motor is connected in between the 2 NMOS transistor in series.



4

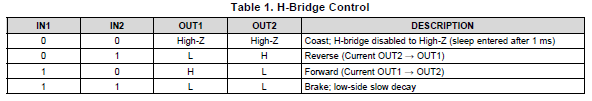
3

2

1

1. Full H Bridge Typography

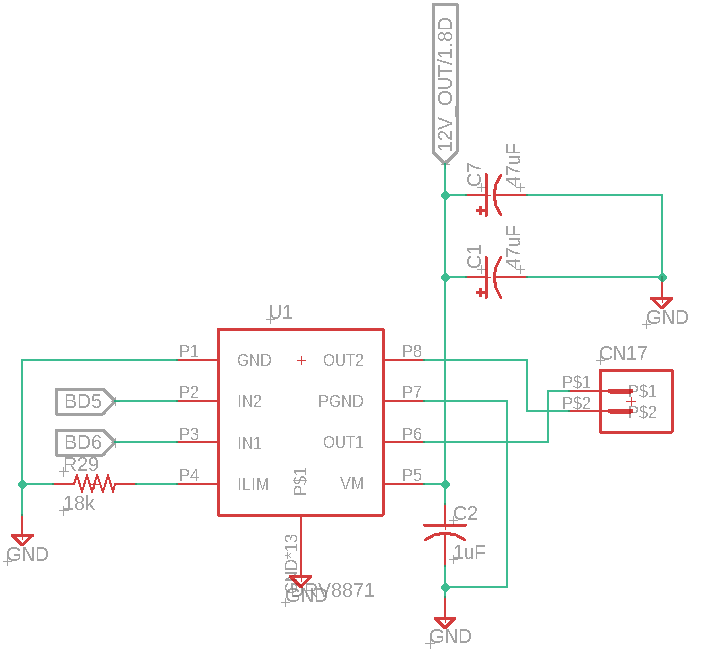
To select the direction of the motor the control circuitry chooses either NMOS transistors 1 and 3 or NMOS transistors 2 and 4. The pairs of NMOS transistors allow the control lines to be reduced from four pins to two pins. Sending a pulse width modulation signal to the NMOS transistors will not only control the direction of rotation of the motor but also speed of rotation of the motor. The average voltage of the signal corresponds to the average voltage out to the motor thus setting the speed of rotation of the motor. The motor driver datasheet has logic table which corresponds with the expected outcomes of various inputs to the control pins on the motor driver.



1. Motor Driver Logic Table

The requirements for motor driver were set by the requirements of the pump 24 V and 1.5 A. We set the requirements of the motor driver to 200% of the current requirements to give headroom for current draw of the motor. Therefore, the motor driver chosen was the Texas Instruments DRV8871 motor driver which supports voltages of up to 45 V and 3.6 A. The DRV8871 has several important features that protect the integrated circuit: overcurrent protection and thermal shutdown. The overcurrent protection is set by a resistor connecting the pin ILIM to ground. The resistor value is determined by the equation the desired trip current of 3 A. The calculated resistor value was 21 kilo-ohm.

In order to protect against rapid voltage changes due to the power supply needing to catch up to the power requirements of the circuit large capacitors were chosen to intervene in these instances. The values chosen were determined by the DRV8871 datasheet which was 47 uF.



1. Motor Driver Schematic

### Design a I2C Gpio Expander and Supporting Circuitry

## Implementation of Task 2.

…

## Implementation of Task 3.

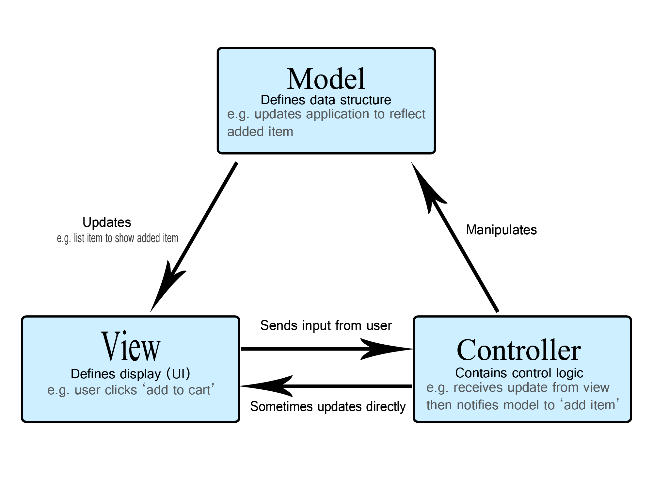
…

## Implementation of Task 4.

### Subtask 1. Design A GUI using Python’s Tkinter Library

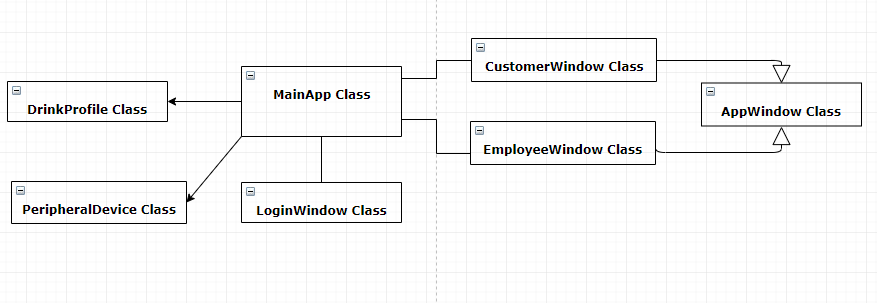
This subtask dealt with designing a Graphical User Interface (GUI) using the Python programming language and its Tkinter library. Python is a high-level scripting language that is used for general purpose programming. Its Tkinter library comes with the initial installation of the Python toolkit, and it offers common GUI components (e.g. buttons, labels, scrollbars). The Tkinter library directly interfaces with the Tk widget toolkit, which is open source, native to various operating systems, and cross-platform. By using Python and Tkinter for the main application, this allows for easy extendibility and portability.

For most GUI applications, the structure of the GUI is separated into different parts. This is usually done for two reasons: the work can be split between different people and there can potentially be more code reuse if the complexity is reduced. These parts are usually the model, view, and control. The model directly manages the data, logic, and rules of the application. The view is any output representation of information. The control accepts input and converts it into commands. This project’s GUI follows this design pattern (also known as MVC).



1. MVC Pattern

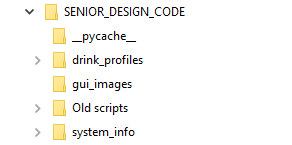
The model would be the MainApp class, which controls the file management of the whole system. The view is defined completely in the CustomerWindow and EmployeeWindow classes. The control is defined in a few different classes because every window needs to report information back to the MainApp object that acts as the central hub. The class structure is shown in a Unified Modeling Language (UML) diagram in the next figure.



1. Class Diagram

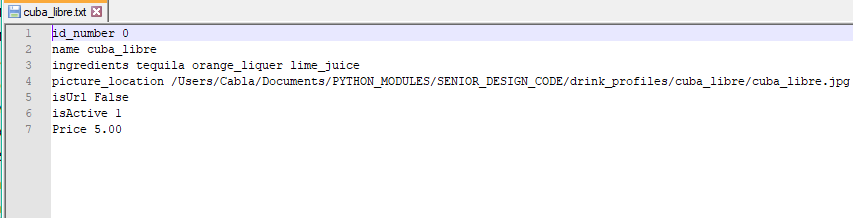
### Subtask 2. Develop an efficient way to organize system & drink information

This subtask dealt with developing an efficient way to organize system and drink information that would be used by the main application, phone application, and peripheral devices. The need for an organized collection of data is crucial because mismanaged information can ruin the performance of the complete system and lower user satisfaction regarding the full product. All the system and drink information are stored in the local file system of the main computer. For easy access, this information is stored in child directories to the “SENIOR\_DESIGN\_CODE” parent directory. The system information is stored in its own directory, “system\_info”, and the drink information is stored in the “drink\_profiles” directory.



1. Parent and Child Directories

Within the drink profile directory, each drink that is offered on our current menu has its own child directory. Each of these children directories have their own image and text file. The images are all the JPEG format because that is the easiest format to work with when using Python’s native libraries for performing functions on images. The text file contains the information needed to build a drink profile inside the main application. On each line of the text file, there are space delimited values that become attributes for Drink objects. There are lines for the id number of the drink, the name, the ingredients, the location of the picture, and the price. The remaining lines are for internal use within the main application. By standardizing this format for the drink profiles, it allows the main application to load an arbitrary amount of drink profiles and contain all the relevant information in Drink objects that are used all over the code.



1. Example of the Text File Format of Each Drink Profile

### Subtask 3. Add in capabilities for communication between main computer & peripheral devices

This subtask dealt with creating an interface between the main application and other peripheral devices. For this system work, there must be a standard interface for commands and data to be sent through. The main application will be sending out commands to peripheral devices and receiving data that will determine the next state of the whole system. The peripheral devices of the main computer are the camera, the payment collector, the embedded boards, and eventually the field programmable gate array (FPGA). By defining a standard interface, adding even more peripheral devices will not take as much time as writing an individual interface for each new device.

This interface is in the form of the PeripheralDevice class. The PeripheralDevice class setups up basic attributes that the main application will use to command and interact with a peripheral device. Some of the attributes are the name of the device, state of the device, the data buffer, the communication method, and the GPIO pin numbers on the Raspberry Pi that the device occupies. With this information, the main application will have complete information about the device and will be able to effectively use that device. Some methods that the main application will call deal with reporting the state to the main app, sending data to the main app, starting communication with the device and the main computer, and terminating communication. With the use of inheritance, the Peripheral Device class will act as the base class for each individual device class. While each individual device class will have its own methods, the main application will only have to use the methods in the base class. This allows for easy extendibility and prevents unacceptable amounts of coupling between the devices and the main application.

### Subtask 4.

### Subtask 5.

## Implementation of Task 5.

### Subtask 1.

### Subtask 2.

### Subtask 3.

### Subtask 4.

## Implementation of Task 6.

### Subtask 1.

### Subtask 2.

### Subtask 3.

### Subtask 4.

1. Conclusion.

In progress.

Acknowledgement

In progress.

Appendix

1. Component Specs
2. Specs of the Main Computer
3. Specs of the Embedded Board
4. Source Code.
5. Source Code of the Python Graphic User Interface
6. Source Code of the Android Application

References

<https://developer.mozilla.org/en-US/docs/Web/Apps/Fundamentals/Modern_web_app_architecture/MVC_architecture> (used for MVC)