“PHIL”  
Implementation Report

By *DrinkTank*

Aaron Riggs, Andre Manz, Chris Baierski, Wesley Scott

## **Recap**

“PHIL” will be built with several main functionalities in mind, all with user-friendly design as a main priority. The following functions will be listed in order of their importance, with the highest priority appearing at the top. “PHIL” will:

1. Pour a variety of drinks, in their standard ratios
2. Allow the user to select drinks via user interface
3. Allow for easy bottle replacement
4. Monitor and store the details of the drinks that have been poured
5. Include an intuitive way to add and remove drinks from the ‘menu’
6. Have an admin area that allows for backend management
7. Keep track of the amount of drinks made
8. Perform business logic on “PHIL”’s usage (profits made, usage statistics, etc.)

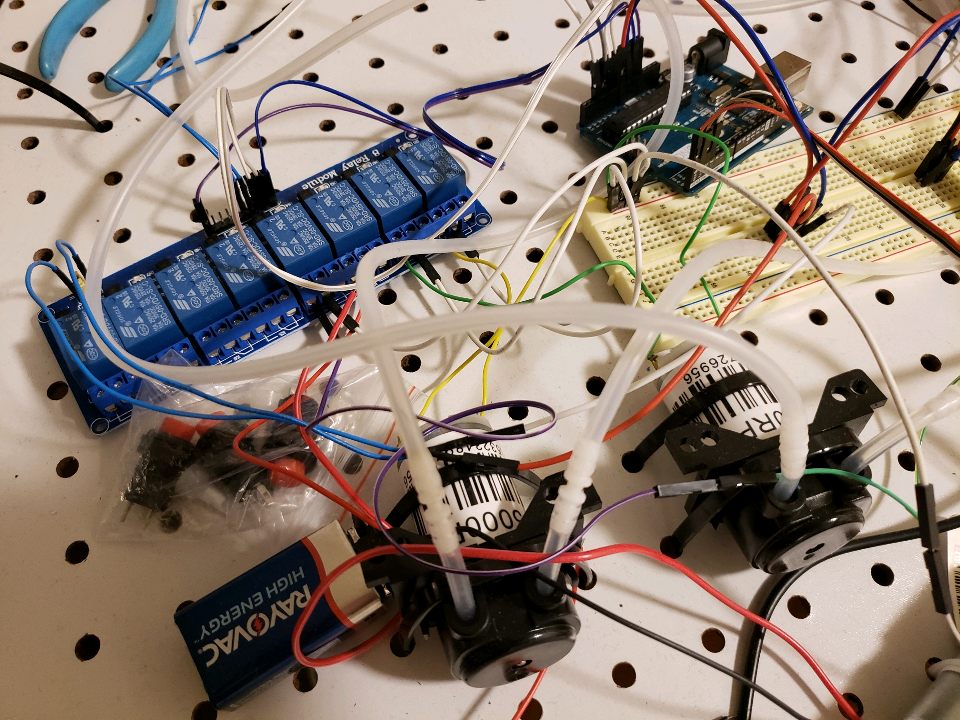
As an automated bartender, “PHIL” clearly needs to be able to pour drinks. “PHIL’s” programming and hardware will be finely tuned to ensure we meet our project’s motivations; the need for the perfect pour ratios every time. In addition, there isn’t much fun in a bartender that makes drinks you don’t ask for. To avoid this, users will be able to order from a list of available drinks. They will select from a screen that shows an intuitive menu. Since we want “PHIL” to last a lifetime, there needs to be a way to easily replace bottles once they have been emptied. This will allow our system to be a timeless addition to any bar, without an inherent expiration date.

As any owner of nearly any business will tell, it is important to know what is happening in your establishment. This is no different when it comes to the drinks industry. Because of that, “PHIL” will keep track of and display information such as what drinks have been poured and their amounts. Also, as “PHIL” moves from location to location, or as new drink recipes are made, there will be a need to add and remove drinks from his repertoire. Finally, we will include an admin area that will allow admin (us the creators, and other selected individuals) to perform checks and maintenance on “PHIL’s” systems.

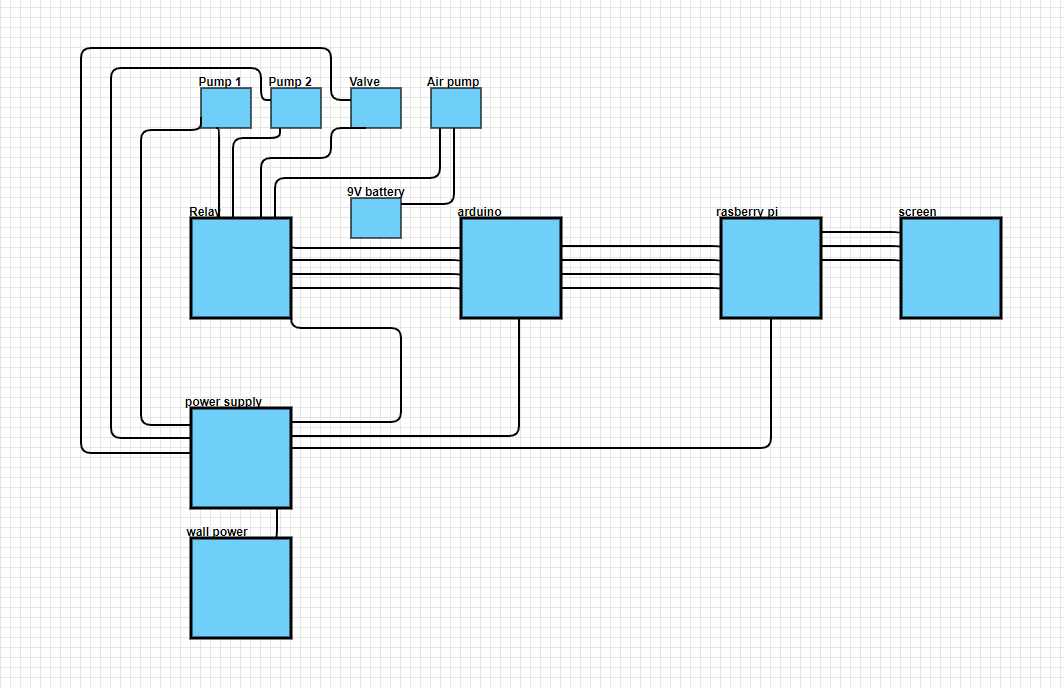
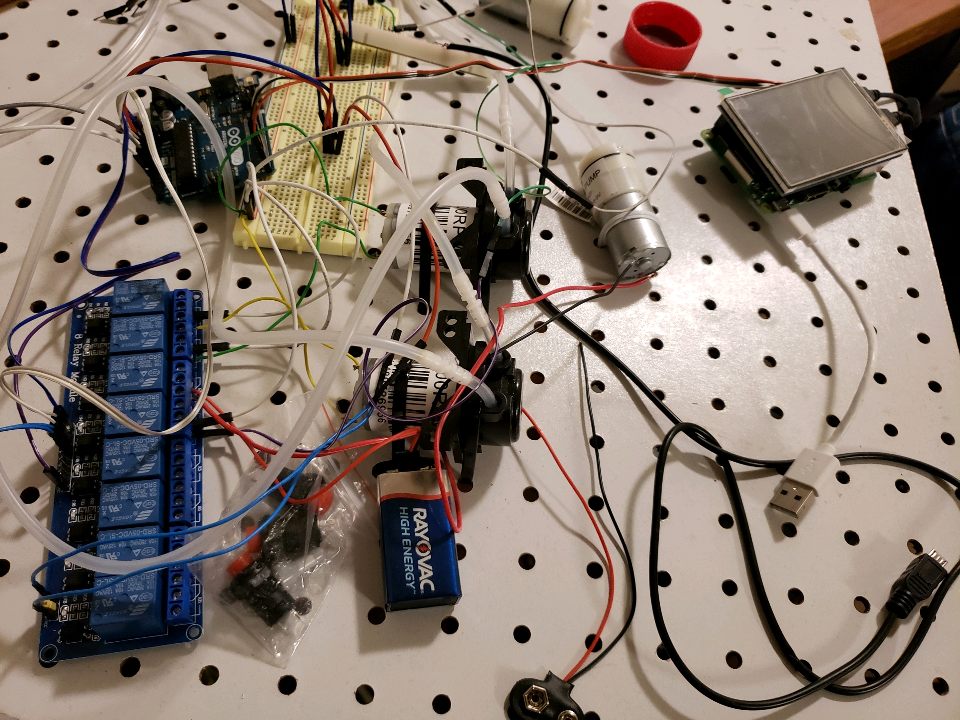
With these functionalities making up the basis of “PHIL’s” systems, we aim to tackle our project’s motivations of helping to reduce pouring errors found in bars across the nation. “PHIL” will be great addition to any restaurant, bar, or party; reducing costs, time costs, and drink inconsistencies.

## **Section 1: RPI breakdown**



The RPi screen with the RPi 3B+ underneath is attached by a jumper that allows communication between the touch screen and the RPi, this allows touch input and GUI output to the screen. The RPi also has a few cables running out of the device to the arduino as you can see in the next image, the arduino directly controls the relays. 

The relays control the pumps, this is done at 12V and 9V through different power sources. The main rails run from the power supplies through the relays and into the pumps to make the liquor/soda flow.



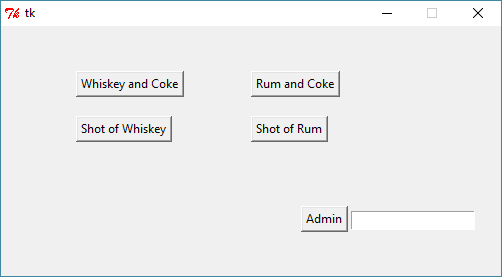
The pictures above are overviews of the rest of the system, you can see that we had to seal a modified cap for the soda bottle for pressurization and how the wiring is done but that is more clear in the system diagram just here.

## **GUI**

The Tkinter GUI has long since been embedded as a part of Python. By simply importking the Tkinter package, users can access a full set of graphical user interface widgets. The widgets in the Tkinter package widgets act as object-oriented classes that naturally have their own unique properties and actions.

Since it is native to Python, Tkinter is a fast and smooth option for graphical user interfaces. However, one of the most noted downsides is that it has an outdated “look and feel”; though for the scope of this project, that is not the biggest of concerns. We chose Tkinter due to how simple it is to import, it’s reliability, and responsiveness. These properties will allow us to satisfy out functional and non-functional requirements without too much of a hassle.

Python Software Foundation. “Graphical User Interfaces with Tk.” *Graphical User Interfaces with Tk - Python 3.7.1 Documentation*, 25 Oct. 2018, docs.python.org/3/library/tk.html.



(A working and fully functional, if a bit simplistic, version 1.0 of our GUI)

In general, graphical user interfaces work by having a main frame or window running a “main loop” that is constantly listening for events. Events (sometimes referred to as actions or commands) occur when a user interacts with a widget/component. This ranges from a user clicking a button, or typing into an entry field.

For the algorithm we are detailing, the pouring of a drink, the graphical user interface acts as the initiator. The user clicks on one of the drink buttons (depicted in the screenshot above) which triggers the event function for pouring a drink. This function interfaces with the hardware in order to turn on and control the pumps necessary for pouring the drink in question.

## **DB**

PHIL implements a lightweight SQLite database. This database selection integrates well with the design PHILosophy of the device. Because we want to remove any external dependencies from PHIL, SQLite allows for a low resource intensive backend. Since the data being stored and queried on PHIL is reduced to just sales information and drink configurations, Cloud storage and a heavyweight SQL Server or the like is not necessary.

Installation was simple:

1. Download and install a version of SQLite3 on Raspberry Pi
2. Initialize a database with the command: sqlite3 PHIL.db
3. Then create tables and query with normal Transact SQL commands and SQLite specific commands found at https://www.sqlite.org/docs.html

Interfacing with PHIL’s Python API is also fairly simple with the “sqlite3” Python library. Documentation can be found at <https://docs.python.org/2/library/sqlite3.html>

The schema for PHIL’s database is as follows:

1. drinkConfig(drinkName: varchar(255), liquor1: varchar(255), liquor2: varchar(255), soda1: varchar(255), cost: decimal(2, 2))
2. drinkSales(saleId: integer, drinkName: varchar(100), timestamp: datetime)

This translates into a table architecture as folllows:

Table Name: drinkConfig

Vodka Soda|Ciroc||Club Soda|7.99

Whiskey and Coke|Knob Creek||Diet Coke|4.99

Table Name: drinkSales

1|Whiskey and Coke|2018-10-19 03:39:31

2|Whiskey and Coke|2018-10-19 03:39:32

3|Whiskey and Coke|2018-10-19 03:39:33

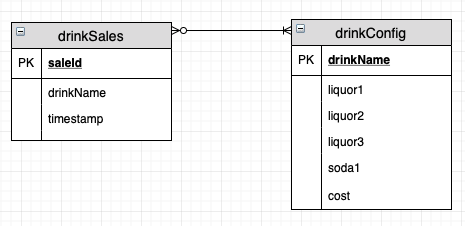
4|Vodka Soda|2018-10-19 03:39:41

5|Vodka Soda|2018-10-19 03:39:42

6|Vodka Soda|2018-10-19 03:39:43

7|Vodka Soda|2018-10-19 03:39:45

8|Whiskey and Coke|2018-10-28 19:51:17



The above ER diagram depicts the current database. The drinkConfig table has a 0 or many relationship with the drinkSales table, and a 1 to many relationship in the opposite direction.

## **Voice Control**

PHIL will make use of the open source Mycroft AI to add voice control functionality. Mycroft AI is a personal digital assistant using natural language processing and machine learning to provide quick and accurate understanding of customer intent.

Mycroft AI makes use of Wake Word Detection. The program will listen but will not record or act on sounds until the Wake Word has been said. This will alert the system that a command will be issued.

The Speech to Text functionality makes use of Google STT. This allows any commands spoken to Mycroft to be translated to text and processed to be used by the built in Intent Parser.

Mycroft Intent Parsing makes use of the text provided by the STT to execute the appropriate functions with the given input. The Intent Parser being used is Padatious, a neural network based system.

Mycroft’s Text to Speech system utilizes Mimic, based on Festival Light. This system allows PHIL to say pre-programmed phrases out loud to the customer/user.

Implementation:

1. First, ensured current Raspbian version is up to date.
2. Using the linux terminal, clone the Mycroft Github repository.
3. Run dev\_setup.sh to install Mycroft.
4. Change wake word from “Hey, Mycroft” to “PHIL”.
5. Develop and import a custom “skill” to interface with the PHIL Python Source code.

Tentative Implementation:

1. Import open-source Mycroft skills to give PHIL the ability to recite drink recipes.
2. Import open-source Mycroft skills to give PHIL the ability to converse with customers.