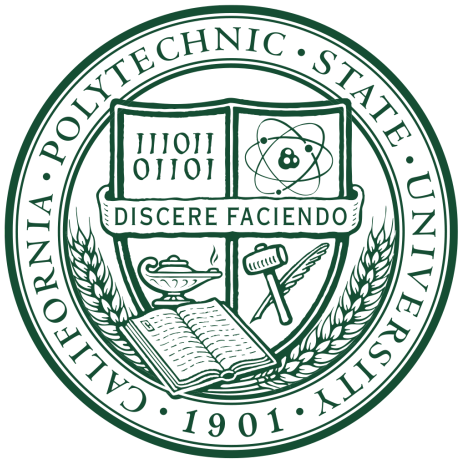
Cal Poly Computer Engineering Senior Project

Joe on the Go

Grayson Meurrens
Nico Ledwith
Advisor: Bruce DeBruhl

Spring, 2017



1 Introduction

1.1 Project Goals

The ultimate goal for this project is to design an automatic, large capacity coffee maker. The system should be able to detect when a cup or mug is in position to receive coffee, then dispense a cup's worth (~ 8oz) of coffee. When the coffee in the urn is getting low, our system should be able to allow a person to put new coffee grounds in the top, then push a button to start the coffee brewing process.

More specifically, we had to accomplish the following tasks in order to meet our goals: solving the issue of transporting water from a reservoir to a coffee urn through a heating component so that it can brew coffee grounds; figuring out how to measure how much coffee has been dispensed by the urn to keep each pour at a consistent 8 fluid ounces; detecting when a cup or mug is in proper position; and keeping track of the amount of coffee left in the urn at all times so that the system can know when more coffee needs to be brewed.

1.2 Project Objectives

In order to meet our goal of having an automated coffee maker, we had to integrate many hardware components. Many of these had to be controlled by a central microprocessor. This system consisted of a large container to hold water, a heating component, and a coffee urn to keep large amounts of coffee warm. The heating component can provide the transportation to the coffee urn through some interesting physics, so we as designers just need to position it correctly. Two liquid flow meters were used to measure the flow into and out of the coffee urn, giving us enough information to determine when new coffee needed to be brewed. A sonar sensor placed under the coffee dispenser provided distance information to detect when a cup or mug is in proper position. Finally, a water valve controlled coffee flow out of the coffee urn to provide consistent 8 ounce pours.

1.3 Project Outcomes and Deliverables

Outcomes and deliverables of this project consist of:

- An all-in-one, automatic, large capacity coffee maker, that only requires human interaction when the coffee grounds need to be replaced.
- A technical manual describing the end-to-end process of how water becomes coffee in our system.

2 Background

2.1 Julian's in Kennedy Library

During the 2015-2016 academic year, Julian's coffee stopped allowing customers to pour themselves coffee. The main issue was that people were stealing coffee. All a customer needed was a cup, and they could just walk up to the coffee urn, pour themselves coffee and leave. Additionally, some workers would just hand out free or reduced price cups, which the customers would use to pour themselves coffee. Now, Julian's requires the worker to pour the coffee themselves, then give it to the paying customer.

On a related note, for customers just buying a single cup of coffee, the line at Julian's can be a huge deterrent to providing Julian's with business. We thought that there must be a

better way to get people coffee without waiting in line. We concede that if a customer wants a latte, or espresso, or anything *other* than plain coffee, then waiting in line is appropriate. But for a cup of brewed coffee, an expedited version would surely be an easy solution. Thus the idea for an automated, large capacity coffee maker emerged. A fully imagined product would take credit card payments, venmo, and ~~even~~ cash, then dispense a specified amount of coffee to the customer. Customers could pay this product, walk own cup or mug, and walk away with a full cup of coffee. Think of the automatic filtered water dispensers across campus, then add coffee and payment. The fully imagined product is quite a bit of work, so for this senior project, we set our sights simply on getting a working version of all the physical components and left the payment system for the future.

2.2 Bubble Pump

Through research, we discovered that a traditional drip coffee maker utilizes a phenomenon called a gas lift, or "bubble pump," to pump hot water up a tube in the machine into the coffee grounds. Cold water sits in the reservoir of a coffee maker, with a tube going out leading to the heating element, a U shaped metal tube connected to 120V to heat up the water. A process flow diagram is shown below, providing a visual representation of this pump system.

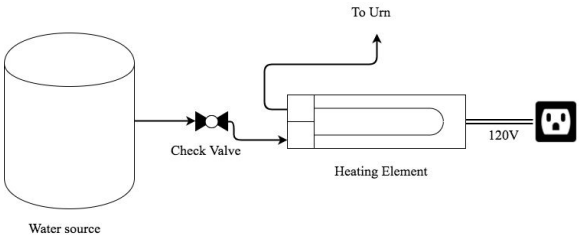


Figure 1: Process Diagram for Bubble Pump

In the tube leading to the heating element is a check valve, shown below. This valve is a simple ball in a socket that allows to water to move one direction only: through the heating element in this case. Water pressure in the reservoir will push water through the valve and into the heating element. Another tube leads out of the heating element and into the coffee grounds sitting above the carafe. But water pressure alone is not enough to pump the heated water into the coffee grounds. Here is where the bubble pump comes into play. When the water is heated in the heating element, the cool water turns into water vapor and hot water. This mixture of vapor and hot water is moved up the tube in two ways. First, the mixture is less dense than the cool water, so the cool water naturally pushed the hot water up the tube (the one way valve prevents any backflow). Second, the water vapor also only have one way to go, so they flow up the tube as well and help move the hot water out of the heating element. [1]

Figure 2: Check Ball Valve [2]

3 Engineering Specs

3.1 Use Cases

To help us in the design process, we created what we thought would be the typical use case for the device. Through this use case we were able to envision scenarios in which the system would be used, thus giving us better insights when making design decisions. The following paragraph describes the primary use case for the device.

The primary use case for this project is a student or faculty member of a university, wanting to buy a cup of coffee without waiting in a line. The "customer", as we will refer to them, will perform the following actions when using our system: set cup or mug under the dispenser and into range of the distance sensor; wait while coffee is dispensed, remove cup or mug from under the dispenser. While this is a simple set of tasks for the customer, our coffee system must function in a way that the consumer would expect in order to meet the customer's expectations. These expectations entail certain specifications about how long the coffee takes to dispense, the temperature of the dispensed coffee, the distance between the cup and the dispenser, and the amount of coffee being dispensed. We have developed engineering specifications in order for this use case to meet the customer expectations, tabulated in the next section.

3.2 Specifications

By thinking about customer needs and expectations, we developed a list of engineering specifications, with discrete values and margins of error. Having concrete values and parameters to design for helped us when making many design decisions. The following table lists out our engineering specifications. It includes a column for risk analysis as well.

Parameter	**Specification**	**Units**	**Tolerance**	**Risk**
Water temp when brewing	180	˚F	Min	L
Amount of water poured in urn	300	fl oz	± 15 fl oz	M
Range cup is detected at	6	cm	± 1 cm	M
Amount of coffee dispensed	7.75	fl oz	± 0.25 fl oz	M
Time it takes to dispense coffee	6	Seconds	± 1 s	M
Coffee temp when dispensed	180	˚F	Min	L

Table 1: Engineering Specifications

4 Design Development

The following section will describe the design development of our system. This section will focus on the various hardware components we decided to implement into our system; it starts with a block diagram showing the connections between all components, then goes into detail about each individual one. Throughout, we give a description of why the each component is necessary in our design and provide images of the component.

4.1 Block Diagram

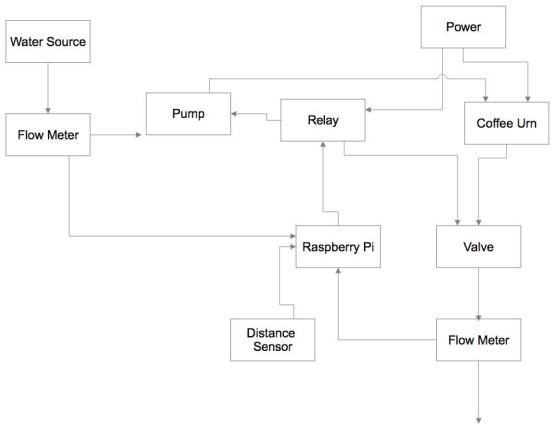


Figure 3: Hardware Block Diagram

4.2 Hardware

4.2.1 Pump

In order to heat up water and push it to the coffee urn we implemented the bubble pump described above. This allows us to have a stationary water source that we could control as well as heat. This is connected to the relay in order to do so.

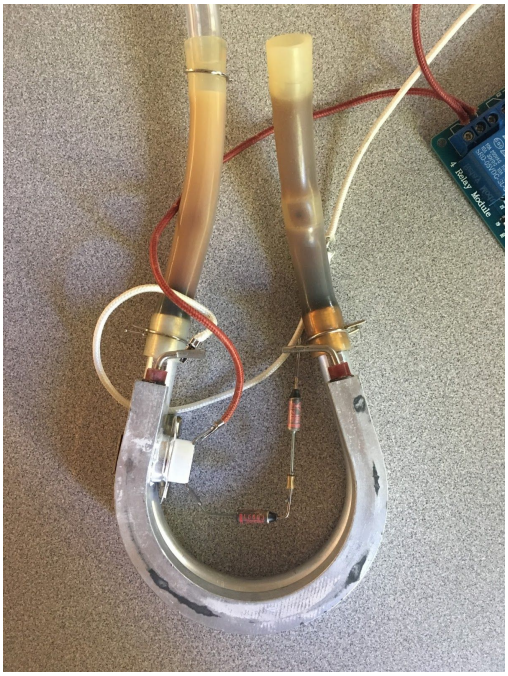


Figure 4: Heating Element / Pump

4.2.2 Relay

For us to control high voltages we have a 4 way mechanical switch relay. This is used to control many parts of the project. The voltages that it controls are the 120V required for the pump and the 12V DC that the valve uses.

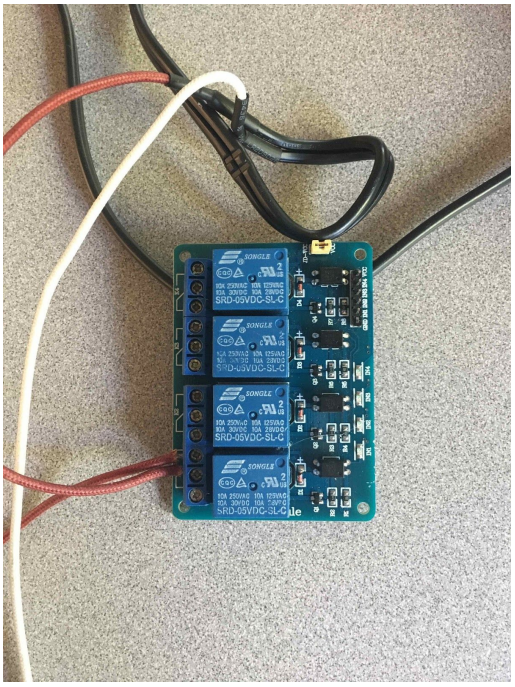


Figure 5: Hardware Relay

4.2.3 Flow Meters

We needed a way to measure the volume of water and coffee we are outputting at certain points in the design. We opted to use two flow meters. These meters work by having an internal wheel that toggles a trigger output every full revolution that it makes. Using this we calculate the average frequency over time and implement some algorithm to finally calculate ounces dispensed. We have one meter that is placed right before the pump so we can tell how much water we are putting into the urn and one that is right after the spout so we can read how much coffee is being poured out.

Figure 6: Flow Meter

4.2.4 Solenoid/Valve

In order to be able to control the coffee flowing out of the urn, we needed some hardware that could either control the built-in spout or act as one itself. Our first idea was to go with a push/pull solenoid. But once we tested it, the solenoid proved to be too weak to actuate the spout and was ruled out. We then opted to go with a plastic valve. However, this proved to also be inefficient since it require a minimum pressure threshold that we would not be able to satisfy once the coffee in the urn reached a certain level. Finally, we moved on to our current brass valve that does not require a minimum pressure to work. This requires 12V DC, so it is also connected to the relay switch.

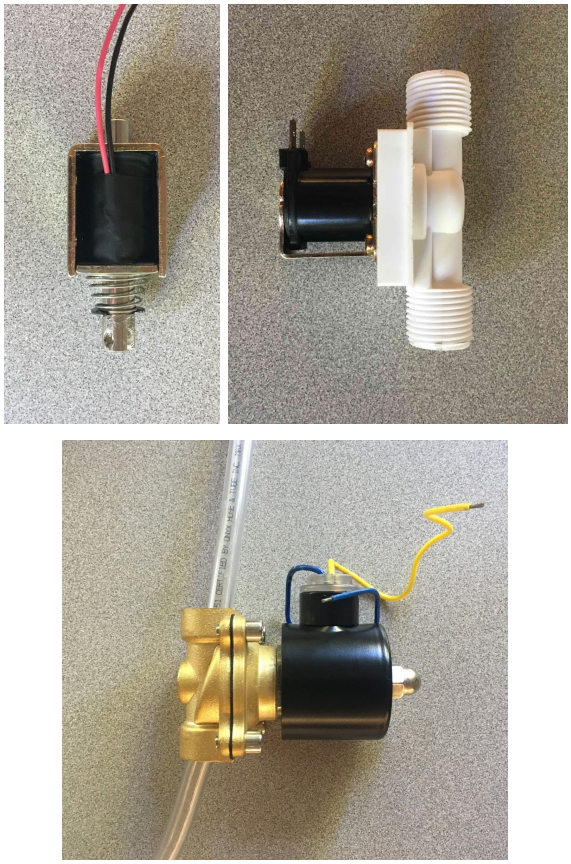


Figure 7: Solenoid, Plastic Valve, Brass Valve (clockwise from top left)

4.2.5 Distance Sensor

We needed a way to detect if the user placed a cup in range of the valve, so the system could tell if it was ready to pour. To do so, we use an ultrasound transducer that only requires 5V to operate. This tells us in real time how far an object is away from it.

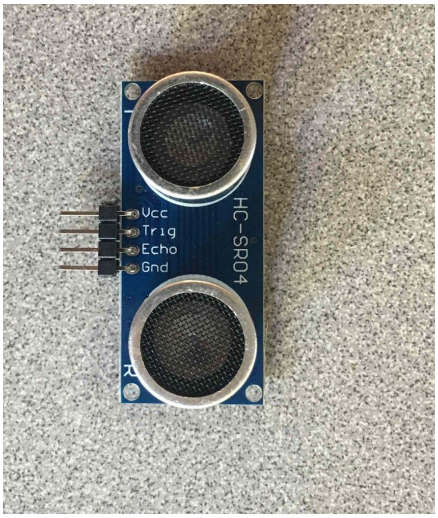


Figure 8: Distance Meter

4.3 Raspberry Pi

To act as the control unit for our project we chose to implement a Raspberry Pi 3 board. This gives us a nice balance of powerful and ease of use. At the early stages of development we ran into a connectivity problem with the Pi. We were unable to ssh into it through WiFi. Our solution was to connect to the Pi through ethernet everytime we wanted to edit our program. The Pi controls the relay and reads from the distance sensor and flow meters. It does so through a Python program.

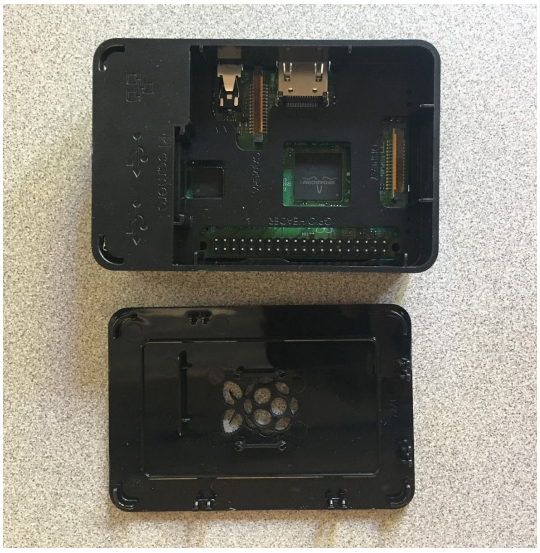


Figure 10: Raspberry Pi (in case)

4.4 Software

The whole system runs on a Python program that implements a state machine that changes states based on the sensors that the Raspberry Pi reads from. The five states that we use are Idle, Pouring, Ready to Serve, Paid, and Serving. By rotating from state to state we are able to maintain a consistent and efficient system. We also have multiple small programs for testing purposes. For each of the hardware components we have a small script to test it works how we expect.

5 Final Concept

5.1 Software State Machine

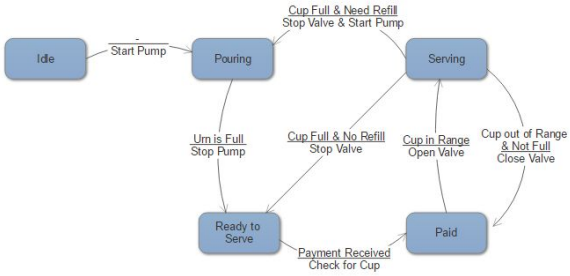


Figure 11: Software Finite State Machine Diagram

Note: We decided to simulate payment instead of implementing a whole transaction system. The payment was simulated through a hardware button press.

5.2 Final Product Cost Estimate

The following table lists each component in our system along with its cost. This table provides a total bill of cost for the whole system. Cost of components was a consideration when designing, as we had a $200 limit.

Item	**Cost**
Coffee Urn	$34
Bubble Pump from Coffee Maker	$9
Water Source	$3
Tubing	$5.78
Raspberry Pi 3 + Kit	$70
Relay Switch	$7
Flow Meter (2)	$19
Distance Sensor	$6.38
Brass Valve	$24
Total	**$178.16**

Table 2: Cost Estimate

6 System Integration and Testing

After we decided which components to implement into our design, we needed to test them. The following section provides details about testing methods for each hardware components. We discuss those that succeeded as well as those that failed.

6.1 Hardware Tests

6.1.1 Pump

- Control Test
  - To test that the pump actuates with the relay switch, we made a program that toggles the relay with a key press. If the pump

turned off/on with the key press, then it passes.

- Volume Loss Test
  - To test that the pump does not lose any of the water that flows throughout, we measured the amount of water entering the pump in fl oz and then the amount exiting in fl oz and compared. If the amount after was within 5% of the amount before, then it passes.

6.1.2 Flow meter

- Volume Test
  - To test how accurate the flow sensors were, we developed a program that keeps a running average of the average frequency of the flow meter. With that average we were then able to calculate the total amount of volume that passed through. We ran this test with a known fixed amount of water to see how accurate that program was. If the test showed the the measured amount was within 1 fl oz of the actual amount, it passed.

6.1.3 Relay

- Control Test
  - Same test as the control test for the relay.

6.1.4 Distance Sensor

- Precision Test
  - In order to test if the distance sensor could detect objects from a relatively close distance we developed a program that continuously outputs the reading from the sensor. We then moved a cup closer and closer to the sensor and if the sensor was able to pick up the object from 6 cm or less, it passed.

6.1.5 Solenoid - Failed

- Push test
  - To see if the solenoid was sufficient to actuate the spout of the urn, we applied the correct voltage (20V DC) to the solenoid and positioned it right above the spout. Since the solenoid was not strong enough to push it, the test failed and we had to move onto another option.

6.1.6 Valves

- Flow Test
  - Since the solenoid failed, we had to figure out a new way to control the flow of coffee out of the urn. We opted to go with a valve. To test the valve we used the control test explained above and attached it to the urn. Once the voltage was applied, the valve passed if coffee was able to flow out. The first valve we had was a plastic one that required a minimum amount of water pressure. This valve failed since the pressure threshold was not being met. The next valve was brass and did not have a minimum pressure requirement so it passed.

6.2 System and Unit Tests

For the overall system we wrote several test cases to check the functionality of the project. Unit tests were also performed on each of the above hardware components to ensure they functioned as we expected. Each of these tests consisted of a short Python script to control and / or measure the component. Manual physical testing and validation was performed in conjunction with each script. Once each component was tested, we combined them all into our system and tested it in a similar fashion. Using our main Python script, we ran our system and manually tested and verified that certain behaviors were ones we expected. This included turning the pump on when the coffee level became too low and dispensing the correct amount of coffee each time via the brass valve.

7 Division of Labor

7.1 Teammate 1 - Nico Ledwith

- Backend Hardware
    - Bubble Pump
    - Flow Meter 1
    - Relay Switch
- Mechanical Setup
    - Coffee Urn
    - Tubing
    - Wiring
    - Mounting Raspberry Pi
- Corresponding Software Design and Testing

7.2 Teammate 2 - Grayson Meurrens

- Frontend Hardware
    - Valve
    - Flow Meter 2
    - Distance Sensor
- Raspberry Pi Setup
    - Installing OS
    - Connectivity Issues/Solution
- Corresponding Software Design and Testing

8 Appendix

8.1 Python Code

```
"""
Joe on the Go
Nico Ledwith and Grayson Meurrens
Cal Poly Computer Engineering Senior Project
"""
#enable GPIO on Raspberry Pi
import RPi.GPIO as GPIO
#activate Broadcom-chip pin numbers
GPIO.setmode(GPIO.BCM)

#Set state constants
IDLE = 1
BREW = 2
READY = 3
PAID = 4
SERVING = 5
CUP_SIZE = 8   #cup size in fl oz
URN_SIZE = 300 #urn size in fl oz
#Setup data
pinSetup()
state = 1

while True:
  if state == IDLE:
    #press c to start
    if key == 'c':
      state = BREW
  elif state == BREW:
    #once urn is full, state is ready
    togglePump()
    if urnLevel() >= URN_SIZE:
      togglePump()
      state = READY
  elif state == READY:
    #check for payment
    if hasPaid():
      toggleDist()
      state = PAID
  elif state == PAID:
    #if customer paid, then wait for cup to be in position
    distance = checkDist()
    if distance < 6 and distance > 4:
      toggleValve()
      state = SERVING
```

```
  elif state == SERVING:
    #pour unless cup moves or is full
    distance = checkDist()
    level = amtPoured()
    if distance > 6 or distance < 4:
      toggleValve()
      state = PAID
    elif level >= CUP_SIZE:
      toggleValve()
      #check if urn needs to be refilled
      if urnLevel() < 2 * CUP_SIZE:
        state = BREW
      else:
        state = READY
  else:
    print("Invalid State\n")


"""list of functions
  pinSetup()
  togglePump()
  urnLevel()
  hasPaid()
  toggleDist()
  checkDist()
  toggleValve()
  amtPoured()
"""
```

8.2 Configuration

Via https://www.raspberrypi.org/forums/viewtopic.php?f=66&t=18207:

- On the Raspberry Pi we entered the bash command:
  - `sudo apt-get update`
  - `sudo apt-get install libnss-mdns`
- Then we connected our Mac to the Pi via a Cat5 Ethernet cable
- The Pi was available to `ssh` into via `ssh -X pi@raspberrypi.local`

8.3 Sources Cited

[1] http://petrowiki.org/Gas_lift

[2] https://cdn.instructables.com/FJM/8GU0/HAQ3A9GZ/FJM8GU0HAQ3A9GZ.MEDIUM.jpg?width=614

8.4 Brief Technical Guide

Brief Technical Document Describing End to End Process of *Joe on the Go*

Grayson Meurrens
Nico Ledwith
Cal Poly, SLO
CPE Senior Project
Spring 2017

This document serves as a technical guide to describe the function of each component in the *Joe on the Go* system. These descriptions will be facilitated through a description of the various states our system goes through in order to turn water into coffee.

1. *Initial state*
    a. Room temperature water sits in the reservoir, a five gallon plastic water dispenser with a vinyl tube attached to the bottom.
    b. The vinyl tube in a) is attached to a flow meter which is attached to a heating element, which in turn is attached to the top of the coffee urn with a second vinyl tube. The heating element is also electrically connected to a physical power relay.
    c. The coffee spigot of the coffee urn is constantly in the "pour" position, but coffee flow is controlled by a brass valve connected to the end of the spigot.
    d. The valve in c) is connected to the same relay as the heating component, as it requires 12V to operate.
    e. Connected to the end of the valve is a flow meter, to measure coffee flow.
    f. A distance meter sits below the flow meter to measure the distance of a cup receiving coffee.
    g. A Raspberry Pi controls: two flow meters, the physical relay, the brass valve, the distance meter, and a button used to simulate a payment being made.
2. *Brewing*
    a. The Raspberry Pi sends a HIGH signal to the pin connected to the physical relay, thus providing 120V to the heating element.
    b. The heating element begins to heat up to around 200 ºF
    c. This heat, combined with a small check ball valve, moves heated water through the flow meter and out of the element, pumping it into the urn.
    d. The Pi begins reading signals sent from the flow meter. Through this equation, $F = 69 * Q(L/min)$, where F is the frequency of signals received by the Pi, Q is the flow rate, and 69 is a derived constant, the number of fluid ounces is determined.
    e. Once the number of fluid ounces reaches the set capacity of the urn, the Pi sends a LOW signal to the physical relay, thus turning the heating element off and stopping the flow of water.
    f. The amount of water pumped from the reservoir is saved for future use.
3. *Ready*
    a. In this state, hot water is either dripping through coffee grounds or sitting in the urn.
    b. The urn is connected to 120V, providing it enough power to keep the brewed coffee at around 180 ºF.
    c. The Pi is constantly polling the distance meter so see if a cup is within range as well as the payment button.

d. If the distance meter reads that a cup is within 6 ± 1 cm for 1.5 seconds, and the payment (simulated by a button press) is received, the Pi will send a signal to the relay to actuate the valve and open it.

4. *Serving*
    a. After the Pi opens the valve, it begins reading the signals from the flow meter
    b. Using the aforementioned equation, the Pi determines how much coffee is being dispensed from the urn.
    c. Once this amount reaches 7.5 oz, the Pi sends a signal to the relay to turn the valve off. 7.5 was chosen because the flow meter has about a 0.25 oz error in measurement.
    d. At this point, the Pi has a decision to make:
        i. If the amount of coffee dispensed causes the amount of coffee in the urn to fall below 16 oz, the Pi goes into the Brewing state.
            1. *Note: at this point, more coffee grounds will need to be manually placed in the coffee urn.*
        ii. Otherwise, the Pi goes back into Ready.